T.C.

HALİÇ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

VISUAL PROLOG PROGRAMI
VE
ZEKİ ÖĞRETİM SİSTEMLERİ

YÜKSEK LİSANS TEZİ

Hazırlayan

Gökhan KARAOSMANOĞLU

Tez Danışmanı

Yrd. Doç. Dr. Murat BEKEN

Haziran 2007
İSTANBUL

T.C.

HALİÇ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRLÜĞÜNE

Bilgisayar Mühendisliği Programı Yüksek Lisans öğrencisi Gökhan Karaosmanoğlu tarafından hazırlanan " **Visual Prolog Programı ve Zeki Öğretim Sistemleri** " adlı bu çalışma jürimizce Yüksek Lisans Tezi olarak Kabul Edilmiştir.

Tez Savunma Tarihi : 21.06.2007

(Jüri Üyesinin Ünvanı , Adı , Soyadı ve Kurumu) :	İmzası :
Jüri Üyesi : Prof.Dr. Ali OKATAN	
Jüri Üyesi : Prof.Dr. Bekir KARAOĞLU (Elektronik ve Hab.ABD Öğr.Üyesi)	
Jüri Üyesi : Yrd.Doç.Dr. Murat BEKEN (Danışman)	

ÖNSÖZ

Bu tez çalışmasında; eğitimin en önemli yardımcılarından biri olan Eğitim Teknolojisi ve giderek artan, bir ihtiyaç haline gelen Zeki Öğretim Sistemlerini incelemeye çalıştık. Çalışmada öncelikle Zeki Öğretim Sistemlerinin önemi vurgulanmış ve giderek artan bilgi teknolojisi alanındaki ihtiyaçlar göz önünde bulundurularak Zeki Öğretim Sistemlerinin temel yapı taşları olan Yapay Zeka, Uzman Sistemler ve programlama dili olarak da Visual Prolog incelenmiştir.

Tez çalışması boyunca benden yardımlarını esirgemeyen, beni yüreklendiren, sorunların çözümünde aktif rol oynayan tez hocam sayın Yrd. Doç. Dr. Murat BEKEN'e, Sayın Prof. Dr. Ali OKATAN'a teşekkür ederim.

İÇİNDEKİLER

T.C.
HALİÇ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
YÜKSEK LİSANS TEZİ

ZEKİ ÖĞRETİM SİSTEMLERİ VE VİSUAL PROLOG PROGRAMI

Hazırlayan
Gökhan KARAOSMANOĞLU

Tez Danışmanı
Yrd. Doç. Dr. Murat BEKEN

Haziran, 2007

ÖZET

"If we understand the human mind, we begin to understand what we can do with educational technology." **Herbert A. Simon**

Bir sınıftaki her öğrencinin hazır bulunuşluk ve kişisel yeteneklerini baz alan, hangi öğretim metotlarıyla daha iyi öğrenebileceğini ölçebilen, öğretim metotlarını öğrenciyi sıkmadan onun seviyesi doğrultusunda multimedia ürünleriyle zenginleştiren, öğrencinin öğrenme zorluklarını hesaplayıp bu doğrultuda kendini geliştirebilen bir sistem düşünün.

İnsanoğlu bilgisayarı 1970'lerden bu yana eğitimde kullanmaktadır. Bilgisayar Destekli Eğitim(BDE), Bilgisayar Destekli Öğretim(BDÖ), sanal sınıflar, uzman sistemlerin uzaktan eğitimde kullanılması bunlardan yalnızca birkaçıdır. Neyi nasıl öğretelim sorusudur bizleri en çok meşgul eden. Sorularla dolu hayatımızı belki de kolaylaştıran, bizim yokluğumuzda yine bizim cevaplarlımızla bu boşluğu dolduran, öğrenmeyi hızlandıran, daha keyifli hale getiren, verimli kılan sistemler giderek klasik öğretme-öğrenme yöntemlerinin yerini almakta, eğitim teknolojisi giderek daha da etkin kullanılmaya başlanmaktadır dünyada.

Çalışmamızda bilgisayar teknolojisinin bu en önemli 2 sorusu; ne öğreteceğiz, nasıl öğreteceğiz; zeki öğretim sistemleri, uzman sistemler, yapay zeka başlıkları altında cevaplanmaktadır.

Zeki Öğretim Sistemleri(Intelligent Tutoring System); "Neyi Öğreteceğini, Kime Öğreteceğini, ve Nasıl öğreteceğini bilen yapay zeka ortak oluşumunda yer alan tekniklerden yararlanarak tasarlanmış bilgisayar programları olarak tanımlanabilir.

Zeki öğretim sistemleri; BDÖ adını verdiğimiz Bilgisayar Teknolojisinin eğitimde uygulanmasının sistemli bir şekilde bilgisayar mühendisliği kullanılarak, yapay zeka, uzman sistemler, yapay sinir ağları gibi çalışma alanlarının da yardımıyla eğitime entegrasyonudur. Bu çalışma planlı, bireyin öğrenme hızına paralel ve her türlü yardımcı teknolojiyi içinde barındıran bir çalışmadır.

Ülkemizde özellikle fen bilimlerindeki akademik başarının, zeki öğretim sistemleri kullanılarak, görsel ve işitsel medyanın da yardımıyla daha üst düzeye çıkarılması sağlanabilir. Tabii burada en önemli bileşen yazılımların kaliteli ve verimli öğrenmeye uygun olabilmesidir.

Anahtar Kelimeler; ZÖS, BDÖ, Yapay Zeka, Uzman Sistemler.

HALİÇ UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
DEPARTMENT OF COMPUTER ENGINEERING
MASTER THESIS

VISUAL PROLOG PROGRAMMMING
AND
INTELLIGENT TUTORING SYSTEM

Gökhan KARAOSMANOĞLU

Tez Danışmanı
Yrd. Doç. Dr. Murat BEKEN

June, 2007

ABSTRACT

"If we understand the human mind, we begin to understand what we can do with educational technology." **Herbert A. Simon**

Think about a system which base student's abilites and student's attendances, which considers which educations is useful, doens't bore students when they learn something, which entertains students with multimedia devices. This system can calculate at the same time learning problems.

Mankind have been using computer since 1970's. We can say as examples; Computer Aided Instruction, Computer Aided Education, virtual classses, experts systems about e-learning. We have very very important problems. This is "What" and "How". We waste our times thinking this questions for years. Education technology helps us about teaching problems and makes easy learning and speeds up learning with multimedia technology. When we use education technology we enjoy more, because we learn with more senses. We use this technology everywhere. In the class, at home and sometimes in the street.

At this work we focus two questions of computer technology. “What we will teach?” and “How we will teach”. We will find the answer with Intelligent Tutoring System, Expert Systems and Artificial Intelligence.

Intelligent Tutoring Systems; can solve the problems “What we will teach?”, Who we will teach” and “How we will teach?” with using artificial intelligence technics.

Intelligent Tutoring System is an adaptation which uses experts systems, artificial intelligence, computer technology, computer engineering, neural network. This system contains a serious plan and this system contains a lot of technology in it.

In our country, especially in applied sciences, we can improve the learning with using this system. We can use also mutlimedia in learning. To improve mathematic and science problems, this is a good solution. Because this system contains visual and auditory components. At the same time, we have to solve the problems about software. If we want a good system, we should provide every useful technology in the system.

Keywords; ITS, Computer Aided Education, Artificiel Intelligence, Expert Systems.

ŞEKİLLER VE TABLOLAR

1. GİRİŞ

Zeki öğretim sistemleri düşüncesi ya da eğitimde kullanılmak üzere tasarlanan makineler düşüncesi 1926 yılındaki Sidney L. Pressey'in icat etmiş olduğu çoktan seçmeli sorulara cevaplar üreten makinesine kadar gitmektedir.(Hartley & Sleeman, 1973). Bu makine kullanıcılara sorular yöneltiyor ve verilen cevaplara anında geri dönütler vererek çalışıyordu. Eğitimciler bu mekanizmayı, öğretim için kullanılacak en iyi sistem olarak değerlendirmişlerdi.(EricThomas student, SDSU Dept. of Educational Technology)

Zeki öğretim Sistemlerini günümüze taşıdığımızda ise birçok argümanla karşılaşmaktayız. Giderek BDE, BDÖ, Uzman Sistemler, Yapay Zeka gibi alanlarınla birlikte kullanılan, teknolojinin bilimselliğe katkısında, eğitim bazında başrolü oynayan bir bilim dalı olmaya başlamıştır. Zeki öğretim sisteminin bilgisayar destekli eğitime sağladığı yararlar göz önünde bulundurulursa ne kadar önemli olduğunun farkına varılabilir. Zeki öğretim sistemi Bilgisayar Destekli Öğretimin son basamağı olarak değerlendirilebilir. Zeki Öğretim Sistemi yaklaşımı ile öğrenci etkinliğinin daha fazla olduğu, öğrencinin kendisini tanıma fırsatı bulduğu, bir öğretmene destek olacak şekilde, öğrencinin hazır bulunuşluk seviyesinin ölçüldüğü, öğrenciyi her alanda yönlendiren bir sistem olarak tanımlanabilir.

Zeki Öğretim Sistemi, eğitimde bilgisayarın kullanılmaya başlanmasından sonra kaçınılmaz bir teknolojik alan halini almıştır. Zeki Öğretim Sistemleri öğretimi daha faydalı hale getirmekle kalmayacak, aynı zamanda bilgisayarın işlerliğini arttıracak ve eğitimde bir devrim yaratacak özelliklere sahip bir etkiye sahiptir. Zeki öğretim sisteminin eğitim için iyi bir alternatif olmakla birlikte aynı zamanda birçok problemi beraberinde getiren karmaşık bir sistemdir. Zeki öğretim sisteminin tasarımının her evresinde eğitim – öğretim teknolojileri, bilgisayar teknolojisi ve yapay zeka bir arada kullanılmaktadır. Tüm bu özellikler sistem tasarlanırken alan bilgisi bakımından tam donanımlı tasarımcılar gerektirmektedir.

Zeki öğretim sistemlerinin en önemli özelliklerinden biri de öğrenciyi değerlendirirken klasik sistemlerinden farklı bir şekilde değerlendirilmesidir. Değerlendirme öğrenciyi gelecek derslere hazırlayan bir değerlendirme olmaktadır. Bu hem öğrenci motivasyonu için hem de dersin verimliliği açısından çok önemlidir.

Öğretim yazılımlarındaki tümel(summative) değerlendirme genellikle öğrencinin verdiği doğru ve yanlış sayısını göz önünde tutarak bu doğrultuda bir sonuç üretmektedirler. Öğrencinin öğretmen veya diğer kaynaklarca öğrenme seviyesinin belirlenmesi süreci değerlendirme olarak tanımlanır. Sadece doğru yanlışa odaklı bir değerlendirme oldukça sığ ve öğrencinin gelişimine paralel olmaktan uzaktır. Bunun yerine Zeki Öğretim Sistemlerine monte edilmeye çalışılan ve öğrenci psikolojisi üzerinde olumlu etkiler bırakan, öğrenciye olumlu dönütler veren, onu motive eden, ses, görüntü, video gibi multimedia unsurlarıyla bezenen değerlendirme daha faydalı olacaktır öğrenci başarısı için.

Bu bağlamda Zeki Öğretim Sistemi, yapay zeka, uzman sistemler, Bilgisayar Destekli Öğretim, Bilgisayar Destekli Öğretim, Zeki Öğretim Sisteminin eğitime katkıları, sınırlılıkları incelenerek, yapılan uygulamalar, öneriler ve sonuçlar ortaya çıkacaktır.

2. ZEKİ ÖĞRETİM SİSTEMLERİ

2.1. Zeki Öğretim Sistemi Nedir?

Zeki Öğretim Sistemi en kısa tanımıyla neyi öğreteceğini, nasıl öğreteceğini ve kime öğreteceğini bilen, yapay zeka ve öğretim teknolojileri kullanılarak oluşturulan bilgisayar programlarıdır. Zeki Öğretim Sistemi ileri öğrenme teknolojilerinin son basamağıdır. Zeki öğretim sistemi Bilgisayar Destekli Öğretime destek olması amacıyla oluşturulmuştur. Görevlerine baktığımızda ikisinin de farklı olduğunu görürüz. Bunun başlıca sebebi, BDÖ'nün görevi bilgisayarı teknoloji olarak kullanmaktır, Zeki Öğretim Sisteminin görevi ise yazılımı öğretim aracı olarak kullanmasıdır.

Bugün ilköğretim okulları ve liselerde giderek yerini almakta olan Bilgisayar Destekli Öğretim(Computer Aided Instruction) yerini zamanla Zeki Öğretim Sistemlerine bırakacaktır. Günümüz koşullarında her kurumun, bilgisayar destekli öğretimin ötesinde, her öğrenciye bir öğrenme asistanı sunması, bütçeleri göz önünde bulundurulduğunda imkansız görülmektedir. Her yeni teknoloji gibi bu teknolojinin de zamana ihtiyacı vardır gelişim, verimlilik ve maliyet açısından.

Zeki Öğretim Sistemi sanal bir öğrenme gibi görünse de sonuçlarına bakıldığında ne kadar etkili olduğu görülecektir etkilerinin. Carniage Mellon Üniversitesinde geleneksel BDÖ sistemleri ile ZÖS'lerini karşılaştıran bir araştırmaya göre, ZÖS'lerinin; öğrenme kalitesini %43 arttırdığı, öğrenme süresini %30 düşürdüğü tespit edilmiştir(Frasson, 1998). Buradaki öğrenme kalitesinin artmasının başlıca sebeplerinden biri öğretimin mikro eğitime doğru kayması, kişiselleştirilesi, öğrencinin yeterlilik düzeyine indirgenmesidir.

1973'te Hartley ve Sleeman'ın Zeki Öğretim Sistemlerinin yapısal durumu hakkında yaptıkları bir incelemeye göre, Zeki öğretim Sistemi üç ana bileşenden

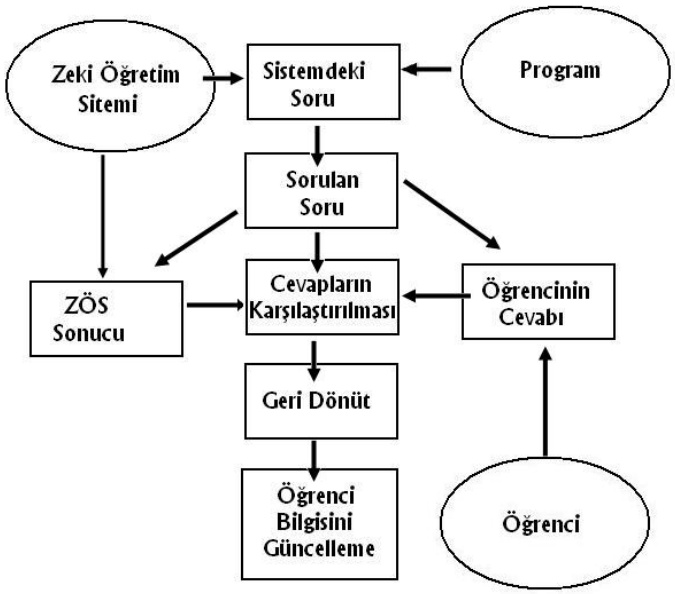
oluşmaktadır.(Hartley,1975) bunlar; bilgi alanı, kullanıcı modeli ve öğretici modeldir.

Zeki Öğretim Sistemi ya da Zeki Bilgisayar Destekli Öğretim olarak adlandırılan eğitime bir alternatif olarak sunulmasına karşın, geleneksel eğitimden daha iyi sonuçlarla karşımıza çıkmaktadır. Sistem sadece sanal bir makineden oluşmamakta, psikoloji, eğitim ve yapay zekayı bünyesinde barındırarak eğitim öğretimin bütün gereklerini yerine getirmektedir.

Bugün uygulamadaki prototipler ve Zeki Öğretim Sistemleri örneklerine baktığımızda, sistemin pratiğe(uygulamaya) dayalı bir yapı içerdiği ve ilköğretim ve liseye dayalı olarak kullanılabileceği görülmektedir. Dahası, teknoloji de yapacağımız uygulamalarda, Zeki Öğretim Sisteminin geliştirilmesi için hazırdır.

2.2. Zeki Öğretim Sistemi Nasıl Çalışır?

Öğrenci bir Zeki öğretim sisteminden sorular sorarak bilgiler öğrenir. Sistem öğrenciye bir soru sorar, verilen cevabı kendi veritabanındaki verilerle karşılaştırarak bir sonuca varmaya çalışır. Sonuçta bilgisayarla öğrenmedir kullanılan teknoloji. Ama öğrencinin burada öğrendiği bilgi yine sisteme kullanıcı tarafından girilen bilgilerdir. Sistemin tek yaptığı kullanıcın verdiği cevaplardan yeni bilgiler türetmektir. Bu da yapay zekanın Zeki Öğretim Sistemlerinde kullanılmasıyla olmaktadır. Zeki öğretim sisteminin çalışma mantığı aşağıdaki şekilde detaylı bir şekilde açıklanmıştır. Şekil Zeki Öğretim Sisteminin çalışma şemasıdır.



Şekil 2.1 Zeki Öğretim Sistemi Çalışma Prensibi

2.3. Zeki Öğretim Sisteminin Yeterlilikleri

a. Öğrencinin hazır bulunuşluk seviyesini ölçebilmeli,
b. Hangi prensipleri, öğretim metotlarını kullanacağını bilmelidir,
c. Bir sonraki adımda ne yapacağına kara verebilmelidir,
d. Girilen bilgileri mevcut bilgileri sağlıklı bir şekilde karşılaştırabilmelidir,
e. Geri dönüt sağlayabilmelidir.

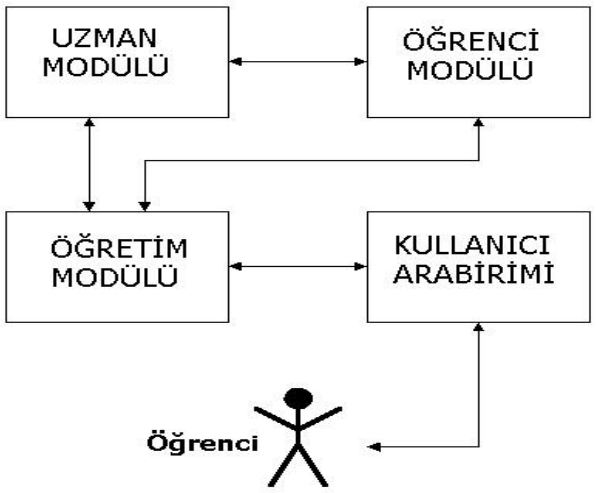
Zeki Öğretim Sistemleri neyi, nasıl ve kime öğreteceğini bilen sistemlerdir. En önemli özellikleri de budur. Fakat bu öğretim sırasında, çeşitli öğretim teknolojileri ve metotlar da kullanmaktadırlar. En önemli iş öğretim metotlarının kullanılmasında yatmaktadır. Zeki Öğretim Sisteminin karar süreci burada önem arz etmektedir.

2.4. Zeki Öğretim Sisteminin Bileşenleri

İleri öğrenme teknolojilerinden olan Zeki Öğretim Sistemi temel olarak dört bölümden oluşmaktadır(Funda Dağ, Kadir Erkan, Zeki Öğretim Sistemleri, ZÖS). Zeki Öğretim Sisteminin alt yapısı oluşturulurken diğer sistemlerden farklı olarak öğrenci, öğretim programı, öğretimde kullanılacak yöntem ve teknikler ayrı ayrı ele

alınarak tasarlanır. Her bir model detaylı bir şekilde tasarlanmak ve teste tabii tutulmak zorundadır sistemin sağlıklı işleyebilmesi için.

a. Uzman Bilgi Modülü(knowledge of the domain)
b. Öğrenci Modeli Modülü(knowledge of the learner)
c. Öğretim Modülü(knowledge of teacher strategies)
d. Kullanıcı Arabirimi Modülü(user interface)



Şekil 2.2 Zeki Öğretim Sistemi Bileşenleri

a. Uzman Bilgi Modülü(knowledge of the domain)

Uzman Bilgi Modülü; problem çözme kabiliyeti olan, uzman bir modülün bilgisayar tanıtımıdır. Sistemin öğretmeye çalıştığı alan bilgisi bu modül içinde yer alır. Dolayısıyla, tüm içerik, öğrencinin öğrenmesi gereken bilgiler, kazanması gereken davranışlar bu modüle dahildir. Uzman Bilgi Modülünün yapması gereken iki ana hedef vardır. Bunlar;

1. İçeriğin, yani soruların, cevapların, güncellenen bilgilerin bulunduğu kaynak görevini üstlenmek,
2. Öğrencinin bilgi düzeyinin öğrenme sonunda değerlendirmesini yaparak öğretimin ne aşamada gerçekleştiğini saptamak.

Bu modül aynı zamanda öğrencinin vermiş olduğu yanıtları baz alarak bilgilerin güncellenmesini de yapmaktadır. Yani yapay Zeka burada devreye girmekte, neyin doğru neyin yanlış olduğunu muhakeme etmekte aynı zamanda, doğruyu ile yanlış yer değiştirmektedir.

Modülün karmaşıklığı ve yapması gereken görev düşünüldüğünde ne kadar iyi bir yazılıma ihtiyaç duyduğu anlaşılabilir. Buradaki programda yapay zeka teknikleri bir sorunun nasıl çözümleneceğini de öğrenebilirler. Mesela bazı Zeki Öğretim Sistemleri, girilen verile doğrultusunda, verileri birleştirerek, bazı kurallar süzgecinden geçirebilir, ve bunun neticesinde sorunları çözebilme kapasitesine erişebilir.

Zeki Öğretim Sistemi sisteminin bu yapısı beraberinde güçlü bir sistem getirir. Çok yönlü eğitim materyalleri ile donatılmış bu Zeki Öğretim Sistemini tasarlamak ise zor ve külfetli bir iştir.

b. Öğrenci Modeli Modülü(knowledge of the learner)

Öğrenci modülü her öğrencinin performansını değerlendirir, bilgi seviyesine karar verir, yeteneklerini ölçer ve öğrenme becerilerinin ne olduğuna sorulan sorular neticesinde karar verir. Burada yapılan asıl işlem, muhakeme yöntemiyle öğrencinin bilgisi, yetenekleri, becerileriyle bilgisayarın hafızasındaki sistem becerilerinin karşılaştırılıp aradaki farkın bulunmasıdır. Buradaki yapay zeka algoritmaları öğrencinin bilgi düzeyini ölçerek uygun yöntemleri uygulamaktadır. İdeal koşullarda Öğrenci Modülü öğrencide karşılaşılabilecek tüm özellikleri kapsamalıdır. Ancak pratikte böyle bir şey imkansızdır. Zeki Öğretim Sisteminde sistem ile kullanıcı arasındaki iletişimi sağlayan teknolojiler fare ve klavyedir. Dolayısıyla yapılacak tüm çalışmalar fare ve klavye doğrultusunda olmalıdır. İnsanın tüm duyularını, mimiklerini, öğrenme becerilerini ölçebilecek bir sistem tasarlamak teorikte mümkün olsa da pratikte değildir. Bunun yerine öğrencinin tüm öğrenme becerilerini ölçecek bir sistem daha mantıklı gelmektedir.

Öğrenci Modelinin Kullanım şekilleri aşağıdaki gibidir:

1. **Düzeltici** : Öğrencinin bilgisindeki hatalarının giderilmesi için bir rehber olarak kullanılır.
2. **Detaylı** : Öğrencinin bilgisini tamamlamada bir rehber olarak kullanılır.

3. **Stratejik** : Öğretim stratejisindeki ana sapmaların düzenlenmesinde yardımcı olarak kullanılır.
4. **Teşhis** : Öğrencinin bilgisinde yer alan hataların belirlenmesinde yardımcı olmak amacıyla kullanılır.
5. **Tahmin** : Öğrencinin öğretim eylemine karşı olası tepkilerinin belirlenmesine yardımcı olmak amacıyla kullanılır.
6. **Değerlendirme** : Öğrencinin ya da Zeki Öğretim Sisteminin değerlendirilmesinde yardımcı olarak kullanılır.

Valerie Shute varsayıma dayanan aritmetik öğretim sistemini, Air Force araştırma laboratuarlında denemiştir. Aşağıdaki örnekte üç öğrenci, kendilerine verilen toplama işlemlerini yanıtlamaktadır.

1. Öğrenci	**22** **+39** **51**	**46** **+37** **73**
2. Öğrenci	**22** **+39** **161**	**46** **+37** **183**
3. Öğrenci	**22** **+39** **62**	**46** **+37** **85**

Şekil 2.3. Zeki Öğretim Sistemi Öğrenci Modeli Örneği

Verilen soruları her üç öğrenci de yanlış cevaplamıştır. Farklı yanlış anlamalar öğrencilerin hata yapmalarına sebep olmuştur. Burada birinci öğrenci taşımada sorun yaşamaktadır. Elde sorunu soruyu doğru yapmasını engellemiştir. 2. öğrenci ise abzen gereksiz olmakla birlikte hep elde sorunuyla karşılaşmıştır. 3. öğrencinin ise basit sayısal ifadelerle ilgili bir sorunu vardır. Örneğimizde her öğrenci verilen soruya bir cevap aramaya çalışmaktadır. Öğretim sistemi ise öğrenciyi yanlış anlaşılmalara karşı uyarıyor. Zeki Öğretim Sistemi her öğrencinin zayıf yanlarını, işlemlerdeki yanlışlıkları, öğrenme becerilerini belirleyerek konuyla ilgili, kişi bazında gerekli önlemleri alıyor

Daha karmaşık örneklerde ise, Zeki Öğretim Sistemi öğrencinin anlamaya ve öğrenmeye yönelik becerilerini bir monitör aracılığıyla kullanıcıya bildiriyor. Zayıf veya eksik yanlarının farkında olan öğrenci bu alanları geliştirmeye yöenlik yaptığı çalışmalarla eksiklerini tamamlıyor.

c. Öğretim Modülü(knowledge of teacher strategies)

Zeki Öğretim Sisteminin öğretim yöntem ve tekniklerine uygun olup olmadığını kontrol eder, pedagojik olarak altyapısını hazırlar. Kullanılacak öğretim yöntemleri, teknikler, verilecek geri dönütler bu modülde belirlenir. Kısacası öğretim materyallerini içerir.

Halen var olan Zeki Öğretim Sisteminde iki farklı sunum yöntemi vardır. Bunlar sokratsal ve çalıştırıcı yöntemlerdir.

Birinci metot olan sokratsal metot; öğrencinin kendi yanlış anlamalarını, eksiklerini gidermeye çalışan öğrenciye bir dizi sorular sorara hedefe varmaya çalışan bir sistemdir. Öğrenci bilgileri sorulan sorular ve verdiği cevaplar sayesinde keşfeder. Bu keşif öğrencinin hazır bulunuşluğuna, ilgi, yetenek ve becerilerine bağlı olarak değişmektedir.

Çalıştırıcı metotta ise; öğrenmenin gerçekleşmesi için bilgisayar oyunları gibi eğlenceli, interaktif ortamlar yaratılmakta, öğrencinin yeni bilgileri öğrenmesi bu ortamlarda sağlanmaktadır. Eğlenerek problem çözen öğrencide bilgiler daha kalıcı olmaktadır. Burada asıl önemli olan konu yazılımın her türlü görsel, işitsel çevre birimleriyle desteklenmesi ve amaca uygun olmasıdır.

d. Kullanıcı Arabirimi Modülü(user interface)

Öğrenciyle Zeki Öğretim Sisteminin iletişimini sağlayan başlıca modüldür. Burada amaç ZÖS'ini oluşturan temel bileşenlerin görsel nesnelerle temsil edilmesidir. Amaca uygunluk açısından kullanımı kolay, anlaşılır olması, sıkıcı, karmaşık olmaması gerekir. Özellikle çalıştırıcı metotta öğrencinin eğlenerek öğrenmesi söz konusu olduğundan, Kullanıcı Arabiriminin öneminin farkına varılmalıdır. Burada kullanıcı arabirimi bir nevi öğrenciyi öğrenmeye hazırlayan, güdüleyen bir konuma sahiptir. Dolayısıyla Kullanıcı Arabirimindeki görsel nesnelerle öğrencinin zihinsel

kapasitesi birbiriyle örtüşmeli, çatışmamalıdır. Ancak bu durumda arabirim ile öğrenci zihinsel etkinlikleri arasında sağlıklı bir işletişi kurulabilir.

Zeki Öğretim Sisteminde arabirimler farklılık gösterebilir. Öğretmen arabiriminden, ders öğretmeni soruları hazırlayıp, öğretim faaliyetlerinin planlamasını yaparken, öğrenci arabiriminden öğrenci kendi seviyesine göre bir ders seçebilir, ya da bu seçimi öğretim sisteminin belirleyeceği sorulara bırakıp, cevaplama yöntemiyle bir seviye belirleme yapabilir. Asıl amaç, öğrencinin kendine uygun bir dersi seçmesidir.

2.5.Zeki Öğretim Sisteminin Ana Karakteristiği

Daha önce de söylediğimiz gibi Zeki Öğretim Sistemi sadece bir bilgisayar yazılımı değildir, tek başına hantal bir makine de değildir, çeşitli öğretim faaliyetlerini bünyesinde bulunduran, yapay zeka yöntemleriyle geliştirilmiş, her bir bölümün ayrı bir model olarak ele alındığı, üzerinde defalarca çalışıldığı bir yapıdır. Zeki Öğretim Sistemi öğretim için sunulmuş iyi bir alternatiftir. Fakat aynı zamanda bu alternatifin, planlanması, hazırlanması, deneme ve test çalışmaları zaman alan, anlaşılması uzmanlık gerektiren karmaşık bir yapıdır aynı zamanda. Dahası Zeki Öğretim Sisteminin planlama ve oluşum aşamalarında öğretim stratejileri, yapay zeka yöntemleri ve her türlü bilgi teknolojisi bir arada kullanılmaktadır.

Şu anda yapılan çalışmalar da göstermektedir ki, BDÖ(Bilgisayar Destekli Öğretim) yerini yavaş ZÖS(Zeki Öğretim Sistemi)'ne bırakmaktadır. Öğretimin zekileştirilmesi beraberinde, verimli ve eğlenceli bir öğretimi de getirmektedir. Bu da yaşayarak, eğlenerek öğrenen ve öğrendiğini kolay kolay unutmayan bir toplum oluşumunun ilk adımıdır.

3. YAPAY ZEKA

3.1.Yapay Zekanın Tanımı

1970'lerde çekilen ve hala insanların gözdesi olan Star Wars sinema filmi Yapay Zekanın belki de en güzel örneklerinden biridir. Ya da Terminatör filminde yarın insan yarı robot, ya da dünyayı ele geçirmeye çalışan robotlar, Cyborglar, Androidler hala gözlerimizin önündedir.

Yapay Zeka terimi ilk defa, YZ konusunda düzenlenmiş ilk konferans olan Dartmouth Konferansında, John McCarthy Yapay Zekanın tanımını yaptı. Fakat yapay zekadan önce tanımlanması gereken şey, zekanın tanımlanmasıdır.

Zeka : Zeka problem çözme yeteneği olarak tanımlanabilir. Ya da bir başka tanımla, insanın algılama, düşünme, muhakeme, yargılama gibi sezgisel becerilerini kullanarak doğadaki olayları çözümleme yeteneği diyebiliriz.

Zekanın tanımına dikkat edecek olursak buradaki kilit kelimenin "insan" olduğunu görürüz. Zeka insana has, insanın problem çözme yetisi olarak görülmektedir. Bu diğer canlıların bir zekaya sahip olmadığı anlamına gelmemelidir. İnsan zekası da kendi içinde değişik safhalara ayrılmaktadır. Çeşitli alanlarda kullanmaktadır insan zekasını. İnsan zekası birimlerle ifade edilemez, fakat bir takım çalışmalarla gelişebilir bir yapıya sahiptir.

Yapay zeka ise, yukarıda saydığımız özelliklere sahip olup, organik(biyolojik) olmayan sistemler olarak tanımlanabilir. Yapay zekaya sahip yapılar, insan gibi düşünmesi istenen sistemlerdir. İnsan gibi karar verip, insanın olmadığı yerylerde karar verme yetisine sahip sistemlerdir.

Yapay zeka çalışmaları özellikle İkinci Dünya Savaşının ardından ortaya çıkmıştır. Makinenin ve savaş endüstrisinin ön planda olduğu savaşın ardından çalışmalar yapa zeka üzerinde yoğunlaşmıştır.

3.2. Yapay Zeka Alanları

Bilim adamları Yapay Zeka alanında araştırma yaparken dört grupta incelemişlerdir yapay zekayı. Bunlar;

- İnsan gibi düşünen sistemler,
- İnsan Gibi Davranan Sistemlerdir,
- Mantıklı düşünen sistemler,
- Mantıklı davranan sistemlerdir.

Yapay Zeka alanındaki ilk çalışmalar yapay zekanın donanım alanında daha ağır bastığından yanaydı. Bu belki de yazılımdaki gelişmelerin ileri düzeyde olmamasından dolayıydı. Daha sonraları yapay zekanın asıl işlevinin yazılımda olduğu, yazılımın gelişmesiyle zeki sistemlerin daha işlevsel hale geleceği fikri ağır bastı.

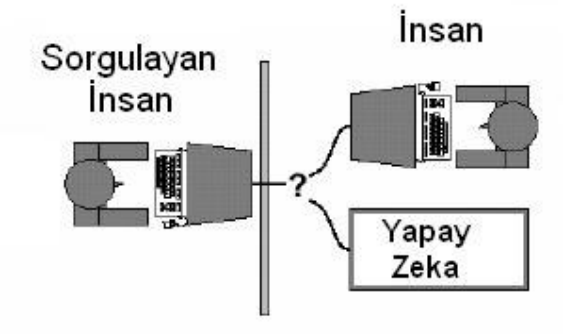
3.2.1. İnsan gibi düşünen sistemler

İnsan gibi düşünen sistemler üretmenin yolu insanın nasıl düşündüğünü, düşünme sürecini, düşünmeyi sağlayan beynin yapısını iyi bilmek gerekir. Bu da yapay zekanın en önemli yardımcılarından psikoloji sayesinde olabilir. Ancak yeterli sayıda deneylerden sonra sağlıklı bir çalışma yapılabilir veya bir kuram oluşturulabilir. Oluşturulacak bilgisayar programı ancak bu kurama dayalı ise sağlıklı olabilir. Oluşturulan programın özellikleri kurama dayalı ise insan gibi düşünen bir sistem olduğu söylenebilir.

İnsan gibi düşünen sistemler üretmek eğitimin, dolayısıyla bilişsel alanın da araştırma konusudur. Burada insan gibi düşünen sistemler tasarlamanın iki amacı vardır. Birincisi, yapay zeka uygulamalarını geliştirmek dolayısıyla bu alandaki açığı kapatmak, ikincisi ise insanın bilişsel süreçlerini yapılan araştırmalar doğrultusunda ortaya çıkarmaktır. Her yeni araştırma keşfedilemez gibi görünen insan beyninin özelliklerini gözler önüne sermektedir.

3.3.Turing Testi

Burada amaç insan gibi düşünen sistemler tasarlamaktı. Sistemin zeki olması insanın zeki olmasından ileri geliyordu. Bir yapının Yapay Zeka olması için yukarıda saydığımız özelliklere sahip olması gerekiyor. Kısacası zeki olması gerekiyor. Bilim adamı Turing 1950 yılında bir makinenin zeki olup olmadığını belirleyen "Turing" testini buldu. Turing testi bir makinenin zeki olup olmadığını belirleyen bir testti. Bir makinenin bu testi geçebilmesi için belirli yeteneklere sahip olması gerekiyordu.



Şekil 3.1. Turing Testi Örneği

Turing testinde amaç zekanın varlığını tespit etmektir. Bu testte denek sorgulayıcıyla bir terminal aracılığıyla haberleşir. Sorgulayıcı deneğin insan mı bilgisayar mı olduğunu anlamaya çalışır. Deneğe bir takım sorular sorar. Denek bu soruları cevaplar. Sorgulayıcı deneğin insan mı yoksa bilgisayar mı olduğunu anlayamazsa denek testi geçmiş sayılır.

Daha önce de değindiğimiz gibi yapay zekada fiziksel(donanım) bölümün pek önemi yoktur. Asıl iş yazılımdadır. Dolayısıyla "Turing Testi" uygulanırken de denek ve sorgulayıcı arasında fiziksel bir temastan kaçınılır. Turing Testinin en önemli özelliği; bilgisayarda zeki davranışı üreten sürecin insan beynindeki süreçlerin modellenmesiyle elde edileceğinin ispatlanmasıdır.

3.4.Yapay Zekanın Temelleri

İnsan gibi düşünen sistemlerden yola çıkılarak geliştirilen Yapay Zeka çalışmaları sadece kendi içinde evrim geçirmemiş aynı zaman da birçok disiplinden de etkilenmiştir. İnsanın davranışlarını inceleyen psikoloji, her türlü teknolojik aracın temeli olan matematik, yapay zekanın donanımının tasarımında etkili olan fizik,

birçok yapay zeka araştırmasına kaynaklık eden dilbilim ve son 50 yıldır büyük bir aşama gösteren bilgisayar mühendisliği bu alanlardan sadece birkaçıdır.

Yapay zeka konusundaki ilk çalışma McCulloch ve Pitts tarafından yapılmıştır. Yapay sinir ağlarına dayalı bu çalışma Turing'in hesaplama kuramına dayanıyordu. Yapay zekanın temeli olan sinir hücreleri bu bilim adamlarının araştırma konusu olmuştur.

1950'lerde Shannon ve Turing bilgisayarlar için satranç programları yazarak yapay zekanın oyun bölümüne geçiş yapmışlarıdı. Yine bu yıllarda İlk yapay sinir ağı temelli bilgisayar SNARC, MIT'de Minsky ve Edmonds tarafından 1951'de yapıldı.

Yapay zeka yavaş yavaş bir teknoloji, bir endüstri haline gelmişti. Şirketler yapay zeka uygulamalarına yönelik araştırmalar yapıyorlar, daha sonra uygulamaya geçiyorlardı. Şu anda bankacılık, sağlık, eğitim, denizcilik, ulaşım gibi birçok alanda yapay zeka uygulamaları kullanılmakta ve şirktetler yapay zeka sayesinde birçok masraftan kurtulup azımsanmayacak karlar sağlamaktadırlar.

3.5.Yapay Zekanın Tarihçesi

Yapay zekanın amacı insan gibi düşünen sistemler oluşturmak olduğundan, bu temel fikir eski yunan medeniyetlerine kadar uzanır. Pratiğe geçirmek yüzyıllar sonra başkalarına nasip olsa da, bu dönemde kullanılan akıllı mekanik araçlar belki de temeli olmuştur bugün kullandığımız araçların.

Leonardo Da Vinci'nin yapmış olduğu çalışmalar ortaçağın karanlığından çıkıp, Pascal'ın geliştirdiği ilk hesap makinesi bir adım daha ileriye götürmüştür teknolojiyi. 19. yy'da ilk programlanabilir bilgisayar geliştirilmiştir.

Sonrasında Turing'in yaptığı çalışmalar, 2. Dünya Savaşı sonrası makineye ve teknolojiye verilen önem doğrultusunda çalışmaların hız kazanmasına sebep olmuştur. Yapay Zekanın temelleri yıllar önce atılmasına rağmen, yapay zeka terimi ilk defa 1956'da kullanılmıştır. John McCarthy Dartmouth Konferansında ilk kez "yapay zeka" terimini bilgisayar terminolojisine sokmuştur.

Sonrasında problem çözebilen bilgisayarlar, yine IBM tarafında üretilen ilk satranç oynayabilen bilgisayar, ve oluşturulan oyunlar neticesinde yapay zekanın bilgisayar oyunlarına da girmesini sağladı. Yapay zeka birçok bilgisayar disipliniyle beraber kullanılmaya başlandı sonrasında. Yapay sinir ağları, bilgisayar destekli öğretim bunlardan birkaçıdır.

Hepimiz Garry Kasparov'a karşı mücadele eden süper bilgisayar Deep Blue'yu bugün gibi hatırlarız. O güne kadar bir bilgisayarın insan zekâsını yenebilmesi imkânsız gibi görünüyordu. Deep Blue , Kasparov'la yaptığı maçta bunun aksini ispatladı. Gerçi karşılaşmaların birçoğu berabere sonuçlanıyordu ama insanın sonuçta biyolojik bir varlık olduğu, bir makine gibi uzun süre aynı sağlamlıkla, hamlelere karşılık veremeyeceği görülüyordu.

İnternetin yaygınlaşması, yapay zekânın daha da yaygınlaşmasını sağladı. Artık yapay zekâ içerikli programlar internet üzerinden milyonlarca kişiye ulaşabiliyor. İnsanlar daha kolay bilgi edinip, bunu uygulamaya dökebiliyor.

İnternet için insanoğlunun belki de en müthiş icadı diyebiliriz. Yüzyıllar süren birikimin ardından insanlığa sunulan bu araç, teknolojini daha da ulaşılabilir olmasını sağlamıştır.

3.6.Yapay Zekânın Çalışma Alanları

3.6.1. Oyunlar: Oyunlar yapay zekânın gün geçtikçe daha büyük bir paydasını oluşturmaktadırlar. Kullanılan teknoloji, özellikle savaş oyunları giderek daha fazla yapay zekâyı kullanmayı gerektirmektedir. Bilgisayar oyunlarında yapay zekânın kullanılmasıyla artık gerçeğe yakın grafikleri görmek mümkün bilgisayar oyunlarında. Karakterler insan gibi yürümekte, yine insanlar gibi etkileşim kurmaktadır. Yine durağan karakterler yerine, plan yapan, monotonluğu ortadan kaldıran karakterler görmekteyiz bilgisayar oyunlarında.

3.6.2. Doğal Dil Anlama ve Çeviri: Dil yapay zekânın gelişiminde önemli bir yere sahiptir. Bilgisayar ile ilişki kurmak için, bir anadilin kullanılması yapay zekânın en önemli alanlarından biridir

aynı zamanda. Bu konuda yapılan çalışmalar şu başlıklarda incelenebilir.

- Yapay zekâ yöntemleriyle tercüme işleminin yapılması
- Metin özetlerinin hazırlanması
- Dokümanların oluşturulmasında yardım
- Kelime işlemcilerin yardımıyla metinlerde yapılan düzeltmeler

Sistemlerin bire bir tanımayla gerçekleştirdiği, eşleme yaptığı dil uygulamaları hala popülaritesini korumaktadır.

3.6.3. Makine Öğrenmesi: Makine öğrenmesini, bilgi düzeyinde, sembol düzeyinde, aygıt düzeyinde öğrenme olarak farklı kategorilere ayırabiliriz. Makine öğrenmesi her üç kategoride de yapay zekâ uygulamalarını gerektirmektedir. Yapılan uygulamalar, bu doğrultuda yapılmaktadır.

3.6.4. Şekil Tanıma: Yaşamın hemen her alanında rastladığımız şekil tanıma yapay zekânın önemli bölümlerinden biridir. Köprü geçişlerinde plakaların tanınmasından, tıp uygulamalarına kadar, göz tanıma sistemlerinden, eğitime kadar çok geniş bir alana hizmet etmektedir.

3.6.5. Ses Tanıma: Özellikle güvenlik ve telekomünikasyon alanlarında kullanılmaktadır. Bugün birçok cep telefonunda bu sistemin basitleştirilmiş biçimini görmekteyiz.

3.6.6. Eğlence sektörü(sinema vs.) : Yapay zekâ uygulamalarını sinemada görmeyenimiz yoktur herhalde. Bu görüntü genelde robotlarla verilse de, insan kılığında robotlar veya basit makineler olarak da karşımıza çıkmaktadır. Bu robotlar bazen dünyayı ele geçirmek isteyen dünya dışı varlıklardır, bazen bilgisayar sistemleri, bazen de uzay gemileri. Yine yüzüklerin efendisi, Star Wars, Troy gibi sinema filmlerinden de göreceğimiz gibi türü önemli değildir sinema filminin yapay zekâ kullanılması için.

3.7.Yapay Zekâ ve Sanal Gerçeklik

Sanal Gerçeklik bilgisayar ortamında oluşturulan bir gerçekliktir. Dünyada cenneti yaratmak gibi bir şey aslında. Cyberspace olarak da bilinen yapay zekânın bu alanında amaç; doğala olabildiğince yakın, bilgisayar arabirimlerinin kullanıldığı bir arabirim oluşturmaktır. Sanal gerçeklik gözlük ve stereo kulaklıktan oluşan başlık seti, vücut hareketlerini algılayan özel bir giysi veya eldivenden oluşan, birçok algılayıcının bulunduğu bir sistemdir. Kullanıcı bu algılayıcılar sayesinde, gerçekle sanal arasında bir yerlerde, nesneleri görebilir veya onlara dokunabilir. Cyberspace, kullanıcının bilgisayar benzetimli varlıklar ile etkileşim içine girebilmesine izin vermektedir.

Cyberspace uygulamalı geniş bir alanda kullanılmaktadır. Akla ilk gelen sistemler, pilot eğitimleri için oluşturulan uçuş simülatörleridir. Üç boyutlu olarak hazırlanan bu simülatörler sayesinde kullanıcı sanki gerçekten uçuyormuş hissine kapılır. Pilot eğitimi için kullanılan bu sistemler aynı zamanda uçma fobisi olan insanların tedavisi için de kullanılmaktadır son zamanlarda.

Sanal Gerçekliğin en önemli kullanım alanlarından biri de CAD adını verdiğimiz Bilgisayar Destekli Tasarım programlarıdır. CAD sayesinde endüstriyel sanal gerçeklik uygulamaları gerçekleşmektedir. Mimarlar ve tasarımcılar ürünlerin üç boyutlu modelleri üzerinde test ve tasarım işlemlerini bu sistem sayesinde yapmaktadırlar. Günümüzde yükse teknoloji kullanılarak yapılan stadyumlar önce bu sistem sayesinde sanal ortamda oluşturulmakta, hatta deneyler bile bu sistem sayesinde gerçekleştirilmektedir.

1998 yılında Dünya Kupası için inşa edilmiş olan Stade De France, önce CAD tasarımı sayesinde bilgisayar yardımıyla tasarlanmıştır. Buradaki amaç eksiklikleri gidermek olduğu gibi, aynı zamanda sağlık ve güvenlik ile ilgili önlemleri de almaktır. CAD sayesinde stadın boşalması aşamasında insan akınları bile analiz edilmiş ve bu doğrultuda önlemler alınmıştır. Bu sistemler alışveriş merkezleri, havaalanları, resmi binalara için de elverişli sistemlerdir.

4. UZMAN SİSTEMLER

4.1.Uzman Sistem Nedir?

Uzman sistemler ancak bir uzmanın çözebileceği türden problemleri çözebilen sistemlere verilen isimdir. Belirli bir alanda sadece o alanla ilgili bilgilerle donatılmış ve problemlere o alanda uzman kişilerin getirebileceği türden çözümlere getirebilen bilgisayar programlarıdır. İyi tasarlanmış sistemler bir problemle karşılaştığında uzman insanların bilişsel süreçlerini taklit ederek probleme çözüm üretmeye çalışırlar.

Yapay Zekâ kuramı 1970'lerin başından bu yana gelişme göstermektedir. Yapay zekâ alanında ilerleyen bilim zamanla yapay zekâya paralel farklı kuramlar ortaya çıkarmıştır. Uzman sistemler de bu kuramlardan biridir. Yapay zekâ alanında yapılan gelişmelere her zaman insan gibi düşünen ve problemlere çözümler üreten akıllı sistemler geliştirme eğilimindeydiler. Uzman sistemler insan becerilerini, yeteneklerini taklit etmeye çalışan bilgisayar programlarından ibarettir.

Uzman sistemler tarafından bilgiye ulaşma birçok aşamadan oluşmaktadır. Bunlar bilgiyi seçme, yapılarına ayırma(çözümleme) ve bilgiyi yönetmedir. Bu unsurlar aynı zamanda uzman sistemin ne kadar güçlü olduğunun da kanıtıdır.

4.2.Gelişim Süreci

Uzman sistemlerin ilk ciddi gelişimi sayılan proje 1965 te E.Feigenbaum ve meslektaşları tarafından Birleşik Devletler Standford Üniversitesinde bir kimyagere, organik bir bileşiğin yapısını, kitle spektrogramının ve ham kimyasal formülünün verileriyle bulması için, yardımcı olmak üzere başlatılan Dendral projesidir. (E.Feigenbaum, The handbook of Artificial Intelligence) Bilgi tabanlı ve uzman sistemlerin asıl amacı, zamanla, verileri ortaya konmuş bir problemi bilgiler ve olgular bütününü kullanarak çözmesi veya çözüm üretmesidir.

Uzman sistemler özellikle tıp alanında gelişim göstermiştir. Dendral projesinin ardından, 1976 yılında Standford üniversitesinde Edward Feingbaum başkanlığında bir grup uzman hekim tarafından MYCIN olarak adlandırılan uzman sistem

geliştirilmiştir. Bu sistem; bakteriyolojik ve menenjit hastalarının tedavisine yöneliktir.

Burada hastanın daha önce geçirdiği hastalıklar, kullandığı ilaçlar, bulgular, muayene sonucunda elde edilen veriler bir araya getirilerek hasta hakkında belli çıkarsamalara varılmaktadır. Sonuçta yapılan tüm incelemeler doğrultusunda, hastanın bir diyagramı çıkarılmakta, sonuç olarak hastaya hangi tedavi yöntemi uygulanacağı, hangi ilaçları kullanacağı, süreç hakkındaki bilgiler ve alternatif planlar sistem tarafından oluşturulmaktadır.

4.3.Temel Bileşenleri

Bir uzman sistem iki bölümden oluşur. Geliştirme ve görüşme çevresi dediğimiz bu bölümler, sistemin mimarisini oluşturmak, verileri sisteme girmek, sistemdeki verilere ulaşabilmek için kullanılmaktadırlar. Bir uzman sistemin ana bileşenlerini aşağıdaki gibi gösterebiliriz;

4.3.1. Bilgi Kazanma : Uzman sistem mimarisinde bulunan bilgilerin sisteme adapte edildiği, aktarıldığı bölümdür. Tüm bilgi kaynakları bilgi kazanma bölümüne kaynaklık edebilir.

4.3.2. Bilgi Tabanı : Sistemin mantıksal bölümüdür. Veriler arasındaki ilişkilerin yapılandırılması, problemler için oluşturulacak çözümler, bu çözümlerin karar yolları bu bölüm tarafından gerçekleştirilir.

4.3.3. Çıkarım Mekanizması: Uzman sistemin en önemli bölümüdür. Bilgi kazanma bölümü tarafından girilen bilgileri, bilgi tabanının yorumladığı yapıları belirli bir metodolojiye göre yapılandıran, problemlere çözümler üreten, bu çözümleri gerçekleştiren, sistem bilgisinin nasıl kullanılacağı hakkında karar veren bölümdür.

4.3.4. Çalışma Alanı: Hafızada bulunan alandır. İşlemlerin ara seviyelerindeki sonuçları kaydetmek için de kullanılır.

4.3.5. Kullanıcı Arabirimi: Sistem yazılımı ile kullanıcı arasında bir çevirmen rolü üstlenerek, aradaki iletişimi sağlayan bölümdür. Çok önemli bir işleve sahiptir. Sonuçta sistem ne kadar iyi olursa olsun

kullanılabilirliği nemlidir. Bu da kullanıcı arabiriminin kalitesine bağlıdır.

4.3.6. Açıklama: Uzman sistemleri diğer istemlerden ayıran özelliklerden biridir. Nedeni ise; kullanıcıya verdiği açıklamalarla onu bir anlamda yönlendirmesidir. Burada amaç; kullanıcıya sistemin kullanılışı hakkında yardım etmek olduğu kadar, aynı zamanda kullanıcının verdiği kararları çözümleyerek, doğru yanlışlarını göstermek, sonuçlar hakkında kullanıcıyı bilgilendirmektir.

4.3.7. Gelişim: Bir uzman sistemin belki de en önemli özelliklerinden biri, verilen bilgiler doğrultusunda, doğru ile yanlışı ayırdıktan sonra bilgileri kendi veritabanına göre güncellemesidir. Burada uzman sistemin yaptığı analiz süreci, bilgisayar öğrenmesi açısından önemli bir süreçtir. Sistemi uzman kılan belki de bu süreçte gerçekleştirdiği performanstır. Bu sürecin sonucunda sistem sonuçlar doğrultusunda iyileştirmelere gidebilir.

4.4.Faydaları

Uzman sistemler artık hemen hemen her sektörde kullanılmaktadır. Aşağıda değineceğimiz gibi sınırlılıkları olmasına rağmen birçok avantajı kullanılmalarını sağlamaktadır.

- **Eğitim: Uzman** sistemler sadece bilgi vermek amacıyla değil, ticari kullanımlarda bile aynı zamanda eğitici bir yere sahiptir.
- **Problem Çözme: Uzman** sistemlerin asıl amacıdır aynı zamanda bir uzman gibi problem çözme.
- **Zamandan tasarruf: Uzman** sistemler uzman insanların uzun sürede çözecekleri problemleri daha kısa sürede, hatasız çözebilmektedirler. Özellikle çok verilerle çalışıldığında insandan çok daha hızlı karar verirler.
- **Güvenilirlik: Uzman** sistemlerin analiz edip çözümlediği veriler defalarca yapılan kontrolörden sonra kullanıcıya sunulmaktadır. Yorulmadan, sıkılmadan, tekrar tekrar gözden geçirilerek harmanlanan veriler ancak sistemin mimarisinde bir sorun varsa yanlış olabilir.

- **Hata giderilmesi: Birçok** uzman sistem yapılan hataların tespiti, öneriler ve iyileştirme için kullanılmaktadır. Bu da toplam kalitenin artması için gerekli bir süreçtir.
- **Verimlilik: Uzman** sistemlerin kullanılması toplam kalitenin artmasını sağlamıştır. Üretim için harcanan tutarlar azalmış, hata giderimi üst düzeye çıkmıştır. Tüm bunlar verimliliği arttırarak kurumların büyük oranlarda karlar sağlamasında etkili olmuşlardır.

4.5.Sınırlılıkları

Uzman sistemler faydaları yanında belirli sınırlılıkları da bünyesinde bulunmaktadır. Amaç bir uzman gibi çalışan sistemler olmasına rağmen, bu istek her zaman gerçekleşmemektedir. Burada sınırlılıklar ticari olarak ele alındığında uzman sistemleri tam olarak çöpe atmak mantıklı değildir. Burada amaç sınırlılıkları zamanla ortadan kaldıracak önlemleri almaktır. Uzman sistemler hiçbir zaman bir insanın yerini alamaz hükmü doğru olmakla birlikte, asıl amaç bir uzman insanın olmadığı yerlerde onun görevlerini yapacak sistem mimarileri tasarlamaktır. Uzman sistemin sınırlılıkları aşağıda sıralanmıştır;

- **Adaptasyon Problemi: İnsanların** uzman sistemlere kolay alışamaması, insan eğitimini makineye tercih etmeleridir. Burada asıl sorun insanlardan sistemin ihtiyaç duyduğu sorulara cevap alamamaktır. İnsanların verdiği cevaplar aynı zamanda sistemin gelişimi için de bir kriter olacağından yeterli sayıda doyurucu cevap alınamamasıdır.

- **Personel sıkıntısı: Uzman** sistemlerin varlığı için bilgi mühendisine ihtiyaç vardır. Sayıları az olan bu mühendislerin bulunması kolay değildir. Aynı zamanda ucuz da değildir. Sistemin verimliliği aynı zamanda bu kişiye bağlıdır. Sistemin maliyeti bu kişiyle beraber artabilir kurumlar için.

- **Gelişim Süreci: Uzman** sistemler bir anda geliştirilen sistemler değildir. Sistemin geliştirilmesi aylar belki de yıllar alabilir. Burada önemli

olan kuruma sağlayacağı faydadır. Doğal olarak, uzman sistem bir bilgisayar programına, belli metodolojik süreçlerin ve yapıların eklenmesiyle, yapay zekânın kullanımıyla oluşturulan bir sistemdir. Bu da gelişim sürecini etkilemektedir.

4.6. Kullanım Alanları

Uzman sistemler artık birçok alanda kullanılır hale gelmişlerdir. Bankacılık, eğitim, otomotiv, sağlık bunlardan sadece birkaçıdır. Her ne kadar belirli sınırlılıkları olsa da uzman sistemlerin esnek olması onları daha da kullanılır hale getirmiştir. Uzman sistemler uzman bilgi sistemleridir. Bu da onları karar verme süreçlerinde yetenekli kılmıştır. Bunun için de karar verme süreçleri açık olmalı ve daha da önemlisi verilen bilgi kesinlikle doğru olmalıdır.

Uzman sistemlerin eğitimde kullanılması ise Mutimedia ürünlerinin eğitime monte edilmesiyle daha da kullanılır hale gelmiştir. Multimedia, Hypermedia, Intellimedia terimleri yavaş yavaş eğitim dünyamıza girmiştir.

Bundan çok değil daha 10 yıl öncesine kadar eğitim materyalleri sadece işitsel ve görsel öğelere dayanıyordu. Intellimedia(Akıllı Medya)'nın ortaya çıkmasıyla artık kullanıcısını tanıyan, onun interaktif olarak iletişim kurabilen, kurduğu iletişim sayesinde bilgi alışverişinde bulanan sistemler geliştirilmektedir.

4.7.Uzman Sistemlerin Eğitimde Kullanılması

Klasik BDÖ(Bilgisayar Destekli Öğretim) uygulamalarının öğrenciyle etkileşime geçebilen sistemler haline getirilmesi uzman sistemler tarafından sağlanmaktadır. Burada uzman sistemlerin bünyesinde bulunan modüller, öğrencinin kişisel özelliklerine göre eğitimin yapılmasını sağlamaktadır. Burada amaç, farklı öğrenci hazır bulunuşluk ve bilgi becerilerine göre sistemin o alana dallanmasıdır.

Özellikle ülkemizde uzaktan eğitim uygulamalarında, eğitimin ülkenin dört bir yanına yayılmasını sağlayacak olan müfredatlar, uzman sistemleri baz olarak oluşturulmaktadır. Açıköğretim sisteminin internet üzerinden yapılan uygulaması

buna en güzel örnektir. Her öğrencinin T.C. kimlik numarasıyla girebildiği sistemde öğrencinin bölümüne bağlı olarak verilen derslerin interaktif bir içeriği bulunmaktadır. Öğrenci buradaki içeriği dinleyebilmekte, okuyabilmekte yine bir geri dönüt olarak, konu sonlarında verilen soruları cevaplayabilmektedir. Cevaplanan sorulara bağlı olarak öğrenci konuyu tekrar etmekte veya bir sonraki konuya geçmektedir.

Yine Microsoft firmasının Milli Eğitim Bakanlığı bünyesinde çalışan öğretmenler için hazırlamış olduğu Öğretmen Eğitim Akademisi uzman sistemler baz alınarak oluşturulmaktadır. Her öğretmenin vatandaşlık numarasıyla girebildiği bu sistem, kullanıcının işlediği dersleri kaydetmekte, tekrar sisteme girildiğinde kaldığı yerden devam etmesini sağlamaktadır. Öğretim interaktif ve aynı zamanda kullanıcının uygulamasıyla gerçekleşmektedir.

İnsan uzmanlığı her şeyin ötesindedir. Yine eğitimde öğretmenin rolü çok önemlidir. Bu rol bir yazılıma veya bir makineye devredilemez. Buradaki asıl amaç, eğitimde öğretmen olmadığı yerlerde oluşturulacak uzman sistemler sayesinde açığı kapatmaktır.

5. MANTIKSAL PROGRAMLAMA, PROGRAMMING IN LOGIC(PROLOG)

5.1.Programlama Dilleri

Prosedurel programlamada programcının sisteme ne yapması gerektiğini söylemesi gerekir. Programcı algoritmayla ilgili her türlü işlemi bilmek ve uygulamak zorundadır. Girilen verilerin dönüştüğü çıktılar da bu algoritmayla ilgilidir.

Deklaratif programlamada ise daha çok tanımlayıcıdır. Sisteme ne yapması gerektiğinden çok biçimle ilgili verilerden bahseder. Burada programcının bilmesi gereken algoritmadan çok bileşenler arasında var olan ilişkilerdir. Deklaratif programlama dilleri aynı zamanda mantıksal programlama dilleridir.

Prolog ise her iki programlamaya izin verir. Mantıksal programlama olmasının yanında aynı zamanda prosedür programcılık da gerçekleştirilebilir Prolog üzerinde. Delphi, Visual Basic gibi programlama dillerinde yapmamız gereken sisteme ne yapması gerektiğidir. İki sayının toplanması da, yaşı büyük olan kişinin bulunması da, bir bankada 1970 yılında doğmuş olanları da bulurken kodlardan yani komutlardan yararlanırız bu tür programlarda. Prolog ise doğruluğu önceden belirlenmiş gerçeklerden oluşur. Kesin olan gerçekler programa tek tek girilir, bunların arasında ilişkiler belirlenir, sistemden output(çıktı) alınacağı zaman bu ilişkiler göz önünde bulundurularak çıktı alınır. Prolog aynı zamanda çözümün ne olacağını değil problemin olduğunu tanımlar.

Prolog programlama dilinde;

- Nesneler hakkındaki olaylar(facts),
- Olaylar arasındaki ilişkiler,
- Nesneler ve ilişkileri aralarındaki kurallar,
- Nesneler ve ilişkileri aralarındaki sorular sorulur.

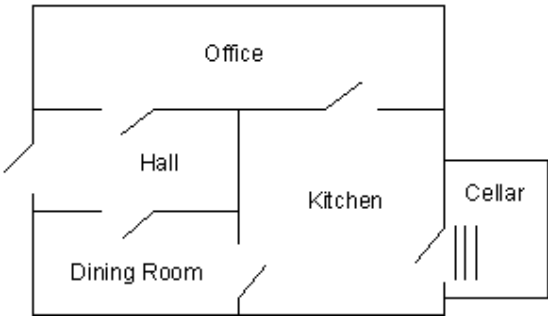
Prolog ilişkisel ve tanımlayıcı olarak tanımlanabilir. Temeli “first order predicate calculus” olan prolog bu özelliği sayesinde daha anlaşılır bir programlama dili olma özelliğini kazanmıştır.

Eski Yunan düşünürü Aristotle (M.Ö.384-322) mantıkla uğraşan ilk kişi olarak bilinir. Syllogistic ya da klasik mantık diye adlandırılan kuramın büyük bir bölümünün yaratıcısıdır. Bugün bile değerini koruyan ve uygulama alanı bulan klasik mantık, bir usavurmalar dizisi sonunda doğru ya da yanlış sonuçlara ulaşma sanatıdır. Bunun tipik bir örneğin,

Kral bir insandır;
Bütün insanlar ölür;
Öyleyse kral da ölür.

biçimindeki bir usavurmalar zinciridir. Klasik mantık, birçok yönden, insan sağduyusunun biçimselleşmişi gibidir. Bu nedenle, konuşulan dile büyük ölçüde bağımlıdır. Öte yandan, hemen her dilde bir sözcüğün farklı anlamlara ya da farklı sözcüklerin eş anlamlara sahip olduğu bilinmektedir. Dolayısıyla, akıl yürütme sürecinde kullanılan sözcükler farklı anlamlara, farklı sonuçlara yorumlanabilir. Bu tutarsızlık, insanoğlunu mantıkta daha kesin yöntemler aramaya itti ve simgesel mantığın yaratılmasına neden oldu.(Prof. Dr.TimurKaraçayakıllı, Veri Tabanları Yaratmaya Doğru)

5.2.PROgramming In LOGic'in Yapısı



Şekil 5.1. Nani Arama Mantığı

Nani arama yöntemiyle prolog mantıksal programlama dilini daha iyi kavrayacağız. Daha önce de belirttiğimiz gibi Prolog mantıksal programlamaya dayanır, yani problemin çözümünden çok ne olduğunu belirler, dahası nesneler arasındaki ilişkileri tek tek belirleyerek bu ilişkilerin nasıl sonuçlar vereceğinin temelini belirler.

You are in the kitchen.
You can see: apple, table, broccoli
You can go to: cellar, office, dining room

> go to the cellar

You can't go to the cellar because it's dark in the cellar, and you're afraid of the dark.

> turn on the light

You can't reach the switch and there's nothing to stand on.

> go to the office

You are in the office.
You can see the following things: desk
You can go to the following rooms: hall, kitchen

> open desk

The desk contains:
flashlight
crackers

> take the flashlight

You now have the flashlight

> kitchen

You are in the kitchen

> turn on the light

flashlight turned on.
...

Yukarıdaki örnekten de anlaşılacağı üzere, Prolog yapısı altında yön bulma tahlillerini de yapabiliriz. Bunun için öncelikle Prolog'ta bulunduğumuz yer tanımlarını yapmalıyız. Burada amacımız ışıkları yakmaktır. Ama öncelikle ışıkları yakacağımız yeri, çevremizdeki eşyaları, eşyaların çevrelerinde bir şey var mı yok mu, ışığı yakmak için nelere ihtiyacımız olduğu gibi sorulara cevap vermemiz gerekiyor.

Öncelikle bulunduğumuz evin odalarını tanımlıyoruz. Amaç programın insan gözüne sahipmiş gibi hareket edebilmesini sağlamaktır.

```
room(kitchen).
room(office).
room(hall).
room('dining room').
room(cellar).
```

Evimizin odalarını tanımladık. Şimdi de odalarda neler bulunduğunu söyleyelim programımıza.

```
location(desk, office).
location(apple, kitchen).
location(flashlight, desk).
location('washing machine', cellar).
location(nani, 'washing machine').
location(broccoli, kitchen).
location(crackers, kitchen).
location(computer, office).
```

Seçtiğimiz eşyalar Prolog programına pek anlamlı gelmese de bizim için anlamlıdır. Amacımız burada ışığı yakmaktır. Işığa ulaşmak için kullanacağımız her eşyayı belirlemeliyiz. Yapılan sadece tanımlamadır burada, yön bulmak için sorgulamaları kullanacağız.

Burada yapılması gereken sadece eşyaların tanımlanması değil aynı zamanda nesnelerin özellikleridir de. Örneğin, oturmak için sandalyenin kullanılması, elmanın yenilebilir olması, eriğin tadının ekşi olması da programa anlatılmalıdır.

Ve son olarak oyuna başlayacak kişinin nerede bulunduğu ve ışığın sönüp sönmediğini kontrol ediyoruz. Nerede bulunduğumuz hareketimiz için önemlidir. Çünkü sisteme tüm tanımlamaları yaparken başlangıç yerimizin belli olması gerekmektedir.

turned_off(flashlight).
here(kitchen).

Prolog dili programlama terimleri, veritabanı verileri ve mantıksal terimlerin bir arada bulunduğu bir dildir. Bir prolog programı tüm yapıları dahilinde bulundurur. Prologun yaptığı daha önce de değindiğimiz gibi problemi çözmekten ziyade problem hakkında bilgi vermektir. Prologu diğer programlama dillerinden ayıran özellik de budur aslında.

5.3.Predicates

Bir ilişkinin sembolik olarak adlandırıldığı isme *Predicate* denir. Daha doğrusu nesnelerin özelliklerinin sembolik olarak ifade edilmesidir. Örneğin bir arabanın fiyatı, rengi, modeli, yılı bu bölümde belirtilir.

car(symbol)
model(symbol)
color(symbol)
male(symbol)
female(symbol)

5.4.Prolog Facts(Gerçekler, olgular)

Prolog mantıksal dil işleme programı facts(olgular)ve rules(kurallar) dan oluşmaktadır. Bu bölümde prologtaki olgulardan bahsedeceğiz. Bunlar Prolog'ta bulundan, basit formlardan oluşan tanımlayıcılardır. Olgular tıpkı günlük hayattaki cümleler gibidir. Noktayla biterler. Olguların bir başka görevi nesneler arasındaki ilişkileri ifade etmesidir. Aşağıdaki örneklere bakarak bunu daha iyi anlayabiliriz.

Örnek 1

Ahmet elma sever.
Mehmet armut sever.
Ayşe kivi sever.

Örnek 2

Ayşe bir kadındır.
Fatma bir kadındır.
Mustafa bir erkektir.
Ahmet bir erkektir.
Ayşe annedir.
Mustafa babadır.
Fatma çocuktur.
Ahmet çocuktur.

Bu ifadeleri prolog dilinde fact(olgu) olarak yazacak olursak aşağıdaki ifadeleri ulaşmış oluruz.

Örnek 1

sever(Ahmet, elma).
sever(Mehmet, armut).
sever(ayse, kivi).

Örnek 2

Kadın(ayse).
Kadın(fatma).
Erkek(mustafa).
Erkek(ahmet).
Anne(ayşe).
Baba(mustafa).
Çocuk(fatma).
Çocuk(ahmet).
Babası(Mustafa, fatma).
Annesi(ayşe, fatma).

5.5.Prolog Queries(Sorgular)

Önce Prolog sistemine tüm özellikleri giriyoruz. Prolog girilen bu özelliklerden bir veritabanı oluşturur. Tüm bu verileri yapısında barındırır. Daha sonra bu özellikler

arasındaki ilişkiler belirlenir. Sonuç olarak Prolog'a tüm bu özelliklerle ilgili sorular sorabiliriz. Soruları sorarken ilişkileri göz önünde bulundurmak gerekir. Prolog ise sorulan sorulara kendi veritabanı dahilinde eğer bu bilgiye sahipse cevap verebilir.

Doğal hayatta soru sorarken aşağıdaki gibi sorarız soracağımız soruyu.

Ahmet elma sever mi?

Bu soruyu Prolog'ta soracak olursak;

Sever(Ahmet, elma).

olur cevabımız. Prolog sorulan bu soruya kendi veritabanına bakarak bir cevap üretir.

yes

Bu örnek her ne kadar çok basit gibi görünse de Prolog Query mantığını açıklamaktadır. Prolog sorulan evet/hayır sorularında aynı şekilde cevap verir. Tabii kendisinde bu bilgi mevcutsa. Prolog kendi kendine soru sorulduktan sonra bilgi üretemez. Bu programın kodlarına bağlıdır. Bilgi varsa cevap verilir yoksa verilmez.

Prolog'un sorgu yapısı sayesinde sorulan her soruya cevap verilir. Soru cevaplanırken, önce verilen olgu(gerçek)ların en başından başlanarak bir tarama yapılır. Ve gerçeklerin en sonuna kadar tarama yapılır. Yapılan tarama sonucunda bulunan cevap kullanıcıya verilir.

Bir örnek verecek olursak ;

What=Ahmet

What=kediler

2 Solutions

Örnekte prolog 2 sonuç üretmiştir. Prolog Ahmet'in kedileri sevdiğini söylemiştir kullanıcıya. Bunu söylemesi için Prolog'un bu bilgilere sahip olması gerekir. O da aşağıdaki kodlarla geçerli olacaktır.

likes(Ahmet, elma).

likes(Ahmet, kediler).

Burada programa Ahmet'in ne sevdiği sorulmuştur. Program da Ahmet'in sevdiği nesneleri veritabanından bularak kullanıcıya sunmuştur.

5.6. Rules(Kurallar)

Verilen olgulardan hangi sonuçları çıkarıyoruz. Ya da olaylar arasındaki ilişkileri nasıl tanımlıyoruz. Kurallar bizi olgulardan diğer olguları ayırmamızı sağlar. Aynı özelliğe sahip nesnelerin karışmaması kurallar sayesinde olmaktadır. Karaların oluşması için, ya da amaca ulaşmak için gerekli yapıları oluşturur. Daha açık ifade edecek olursak; kurallar sonuçları birbirine bağlı değişkenlerin gerçeklenmesini sağlar.

Ahmet mavi olan her şeyi sever.
Ayşe Mustafa'yla ilgili her şeyle ilgilenir.

Yukarıdaki cümleleri prolog programlama dilinde yazacak olursak;

sever(Ahmet, nesne):- sever(nesne,mavi)
ilgi(Ayse, nesne):-ozellik(Ahmet, kisisel).

Yukarıdaki kodlardan çıkaracağımız sonuçlar, Ahmet'in bir nesneyi sevmesi için onun mavi olmasıdır. Ayşe ise Mustafa'nın ilgi alanına giren her özellikle ilgilenmektedir.

5.7.Rules(Kurallar), Facts(Olgular) ve Queries(sorgular) Bir arada Kullanılması

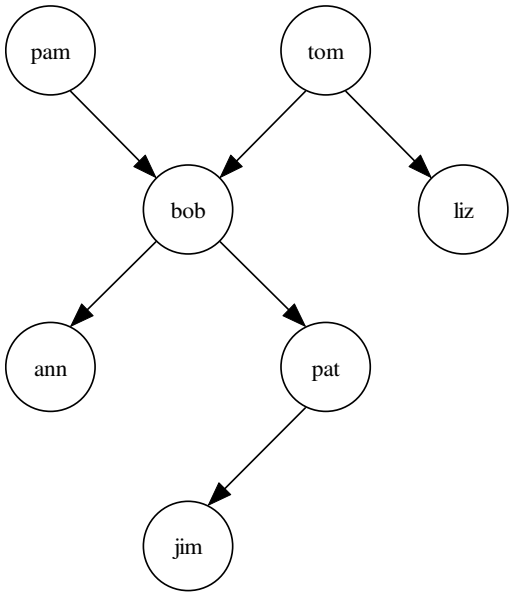
Burada bilmemiz gereken en önemli şey Prolog'un tahmin etme yeteneğinin bulunmamasıdır. Prolog sadece gerçekleri verir bizlere. Aradığı sonuç veritabanında bulunmuyorsa herhangi bir sonuç döndüremez kullanıcıya. Aşağıda örnekte de göreceğimiz gibi Prolog kurallar, olgular ve sorgular bütünüdür. Belli kurallara göre girilen olgular sonucunda bazı sorgulamalar oluşturulur ve program bu sorgulara cevaplar üretir. Bir aile ağacını alacak olursak, bütün fertlerin tanımlanması olguların sisteme girilmesidir. Fertlerin birbirleri arasındaki ilişkiler ve sorgulamalar da sistemin diğer bileşenleridir.

"Tom, Bob'un ebeveynidir" gerçeğinin prolog'da ifadesi :

parent(tom, bob).

Burada "parent" ilişkinin ismidir. Aşağıdaki şekilden de anlaşılacağı üzere her bireyin birbiri arasında bir ilişkisi mevcuttur.

- parent: ilişkinin ismi
- tom ve bob : ilişkinin argumanları



Şekil 5.2. Soy Ağacı Örneği ve Sorgu Mantığı

Burada tüm aile ağacını prolog kodlarıyla gösterecek olursak;

parent(pam, bob).
parent(tom, bob).
parent(tom, liz).
parent(bob, ann).
parent(bob, pat).
parent(pat,jim).

Burada aradaki ilişkileri sorgularken, kim kimin ebeveyni olduğunu sorgulayacağız. Aşağıdaki kodlar bu ilişkileri göstermektedir.

Parent(bob, pam).

Bu soruya programın vereceği cevap "no" olacaktır. GOAL bölümünde ilgili soru sorulduktan sonra program var olan verileri baştan sona tarayacaktır. Veriler arasında böyle bir bilgi olmadığından "no" yanıtını üretecek ve bir pencerede bu yanıtı kullanıcıya iletecektir.

Burada yes/no soruları sorulacağı gibi direkt olarak kim kimin ebeveyni olduğunu ortaya çıkaracak sorular da sorulabilir. Örneğin;

Liz'in ebeveyni kimdir?

ebeveyn(Kim, liz).

Prolog'un cevabı :

Kim = tom

Bob'un çocukları kimlerdir?

ebeveyn(bob, Cocuk).

Prolog'un cevabı :

Cocuk = ann ;

Cocuk = pat ;

no

Kimler kimlerin çocuğudur?

ebeveyn (Ebeveyn, Cocuk).

Ebeveyn = pam

Cocuk = bob;

Ebeveyn = tom

Cocuk = bob;

Ebeveyn = tom

Cocuk = liz;

Buraya kadar yaptıklarımız sadece ebeveynleri belirlemekti. Bunun yanında büyük ebeveyni de(grandparent) belirleyebiliriz. Şekilden de anlaşılacağı üzere soy ağacında birkaç kuşak bir arada gösterilmiştir. Örneğin, Bob Jim'in büyükbabasıdır.

Jim'in büyük ebeveyni (grandparent) kimdir?

Bilgi tabanımızda büyük ebeveyn diye bir ilişki tanımlı değil.

Büyük ebeveyn ilişkisi iki ebeveyn ilişkisinin ‘ve’ lenmesiyle elde edilebilir.

parent(Y, jim):- parent(X, Y).

X=bob

Y=pat

jim’in ebeveynine Y dersek,

Y’nin ebeveyni jim’in büyük ebeveyni (X) dir.

Torun ilişkisi de benzer biçimde tanımlanabilir.

tom, X’in ebeveyni,

X, Y’nin ebeveyni ise

Y, tom’un torunudur.

parent(tom,X):- parent(X, Y).

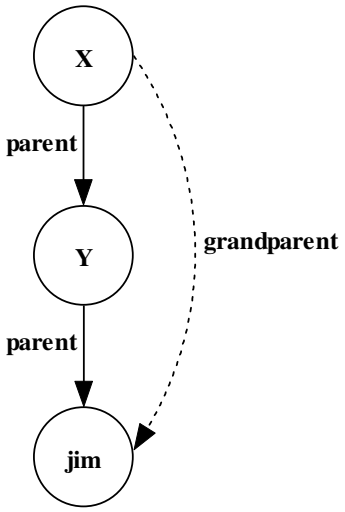
X=bob

Y=ann;

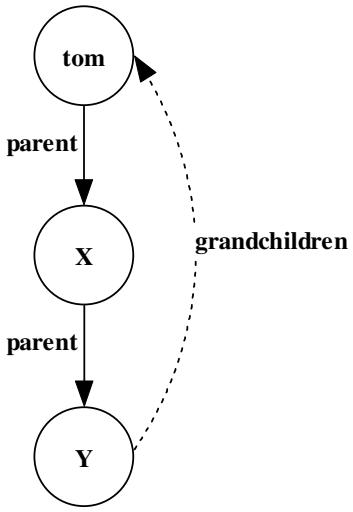
X=bob

Y=pat;

no



Şekil 5.3. Soy Ağacı Dede



Şekil 5.4. Soy Ağacı Torun

Şekil 5.3. te de görüldüğü gibi bir parent aynı zamanda başka bir bireyin torunu olabilmektedir. Bunu belirlemek için yapılan sorguda eğer bir bire başka birinin ebeveyni ise, aynı birey başka bireyin ebeveyni ise son birey ilk bireyin torunudur. Yazılan kodlar da açıklamayı desteklemektedir.

Şekil 5.4. büyükbaba olan özelliği tersine çevirip torunları belirlemektedir. Torun olan birey 2 bireyi de etkilemektedir. Örneklerden de anlaşılacağı üzere, baba olan birey, aynı zamanda başka bir bireyin oğludur. Babası olduğu çocuk da ebeveyninin torunu olmaktadır.

5.8.Değişkenler

Prolog değişkenler sayesinde bizlere genel olguları yazmamızı ve ilişkileri düzenleme imkanı tanır. Değişkenleri doğal olarak programın her yerinde kullanırız. Bizim için çok gerekli olabilirler bazen.

Burada dikkat edilmesi gerek şey değişkenin il harfinin büyük yazılmasıdır. Büyük harfle yazılan değişken programın herhangi bir yerinde kullanılabilir.

Ahmet Aslının sevdiği şeyleri sever.

Prolog'taki kod karşılığı ise şu şekildedir.

likes(ahmet, Something):- likes(aslı, Something).

Something değişkeni büyük harfle başlamaktadır. Bu da değişken olduğunun kanıtıdır.

Gözden Geçirme

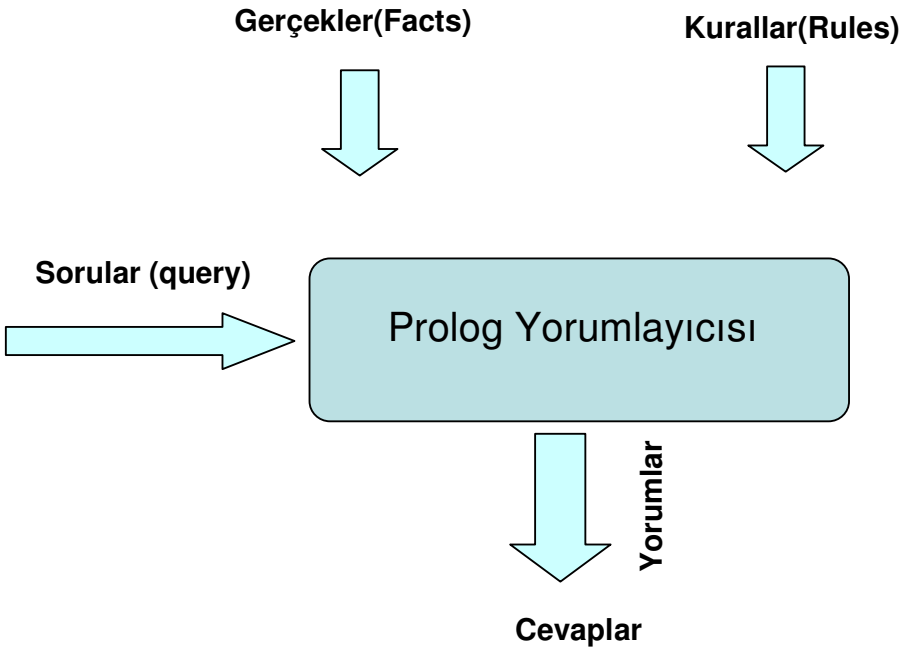
- Bir prolog programı iki bölümden oluşur: Olgular(Facts) ve Kurallar(Rules)
- Olgular nesnelerin programcı tarafından bilinmesi geren sahip oldukları veya birbirleri aralarındaki ilişkilerdir. Bu özellikleri bilen programcı veritabanını buna göre oluşturur.
- Kurallar, bağlı ilişkilerdir. Prolog programlama diline bir ilişkiyi diğerinden ayırma serbestliğini verirler. Bir kural eğer başka bir kuralın gerçekliği kanıtlanmış ise doğrudur. Her kural farklı durumlarının doğru olmasına bağlıdır.
- Prolog'ta her kural iki bölümden oluşmaktadır. Başlık ve gövde dediğimiz bu iki bölüm birbirlerinden **:-** işaretiyle ayrılmaktadır. Başlık doğru olması

gereken, şartı gerektiren bölümdür. Gövde ise Prolog'un doğrulayacağı bölümdür.

- Prolog'ta kurallar ve olgular aynı gibi gözükür. Tek farkları olguların her zaman doğru olması gerekmektedir. Ve tek bölümden oluşmaktadırlar.
- Prolog'un çalışma prensibi girilen birtakım olguların, kurallarla desteklenerek daha sonrasında Query adını verdiğimiz sorgularla bilgilerin süzülmesidir. Program sorgulama işini yaparken önce veritabanının en başından başlar. Daha sonra sonuna kadar tarama yapar ve bilgiye rastladığı yerde kullanıcıya gönderir bilgiyi.

6. VISUAL PROLOG PROGRAMLARI

Visual Prolog mantıksal programlama dili properties(özellikler) ve relations(ilişkiler) yapısı üzerine kurulmuştur. Nesnelerin özellikleri tek tek belirlendikten sonra, programcı tarafından sisteme girilir. Bunların doğruluğu kesindir. Daha sonra nesnelerin özellikleri arasındaki kurallar belirlenir. Nesneler arasındaki ilişkiler belki de programın can alıcı bölümlerinden biridir. Sorgulama(Query) yapılırken bu ilişkiler ön plandadır. Diğer prolog programlarına nazaran Visual Prolog bir derleyicidir.



Şekil 6.1. Prolog Çalışma Prensibi

6.1.Visual Prolog Programının Temel Bölümleri

Genel olarak Visual Prolog dört ana bölümden oluşmaktadır. Bunlar Clauses(Olguler veya Kurallar), Predicates(Yüklemler), Domains(Değişkenler) ve Goal(hedefler)dir.

Clauses(Olgular ve Kurallar) bölümü Visual Prolog programını en önemli bölümüdür. Bu bölüm programcının temel olguları, gerçekleri, aralarındaki ilişkileri, kuralları belirlediği bölümdür. Bu bölümde oluşacak en küçük bir problem daha sonrasında telafi edilemeyecek sorunlara sebep olabilir.

Predicates(Yüklemler) bölümü, sizin eylemler için kullanmak istediğiniz veri tiplerini belirlediğiniz bölümdür. Program akışı içinde kullanılacak değişkenler burada belirlenir.

Domains(Değişkenler) bölümünde Visual Prolog'ta bulunmayan değişkenleri tanımlayıp bu değişkenleri programlarınızda kullanırsınız.

Goals(Hedefler) bölümü ise programın gövde kısmı olup, ilişkileri, özellikleri, olguları belirlemeye başladığınız bölümdür.

6.2.Clauses(Olgular ve Kurallar) Bölümü

Tüm olguları(facts) ve kuralları(rules) yazdığımız bölümdür. Programın derlenmesi sırasında Prolog tüm kodları birinci satırdan itibaren taramaya başlar. Bütün gerçekleri ve kuralları gözden geçirir. Ve bir eşleştirme yapar, aradığı bilgiye ulaştığında ise eşleştirmeyi bitirir ve kullanıcıya geri döndürür. Eğer aradığı veri kümesi istenen mantıklı sonuç değilse, başka bir değeri eşletirmeye başlar. Bu özellik Prolog programının en önemli özelliklerinden biri olan Geriye İz Sürme(Backtracking) özelliğidir.

6.3.Predicates(Yüklemler)

Olgular ve kurallar bölümünde eyleminizi tanımladıysanız veya kullandıysanız bunu Predicates(Yüklemler) bölümünde de yapmalısınız. Visual Prolog'ta değişkenleri ya da yüklemleri kullanırken onları Prolog'a anlatmamamız gerekiyor. Yoksa program bunları derlendiğinde tanımayabilir. Dolayısıyla programın işleyişinde sorun çıkacaktır.

Olgular ve kurallar kullanacağımız yüklemleri tanımlarlar. Yüklemler bölümü kullanacağımız tüm maddelerin bulunduğu bölümdür. Burada yüklemleri belirlerken veri tiplerini de belirlemek zorundayız. Visual Prolog programının en önemli bölümü Olguların ve Kuralların bulunduğu bölüm olsa da Predicates bölümü de veri tiplerinin ve yüklemlerin tanımlandığı bölüm olduğundan en önemli bölümlerden birdir.

Predicates bölümünde özellikler belirtilirken genellikle en az 2 seçenek bulunur. Predicates bölümünde tanım yapılırken aşağıdaki şekilde yaparız.

PREDICATES
Yüklem_adı(argüman_tip_1, argüman_tip_2, argüman_tip_3)

DOMAINS
person, activity = symbol
car, make, color = symbol
mileage, years_on_road, cost = integer

PREDICATES
likes(person, activity)
parent(person, person)
can_buy(person, car)
car(make, mileage, years_on_road, color, cost)
green(symbol)
ranking(symbol, integer)

Örneğimizde arabaları seven bir kişinin hangi özellikleri tercih edeceği gösterilmektedir. Örneğin Ali arabaları sevmektedir. Bunu, *likes(ali,araba)* olarak belirtebiliriz. Yine Ali'nin alabileceği araba modelini ise, *can_buy(ali, hyundai)* olarak gösterebiliriz. Yine arabanın rengini, kişilerin özelliklerini bu bölümde belirtiyoruz.

6.4.Domains(Değişkenler)

Standart programlama dillerinde veri tipleri bellidir. Ve variables dediğimiz bölümünde tanımlanırlar. Daha sonradan değişiklik yapılmayan verilerdir bunlar. Tipleri standarttır. Visual Prolog programlama dilinde de bu aynı aslında. Prolog'ta tanımlamaları Predicates bölümünde yapıyoruz. Değişkenler bize farklı veri tiplerinde çalışma imkanı verirler. Bir Visual Prolog programında, ilişki içinde olan veriler değişkenler bölümüne aittir.

Visual Prolog programı yazılırken bir programcı değişkenlere bağımlı olmak zorundadır. Çünkü yapacağı işlemler, veri kümeleri bunu gerektirmektedir. Değişkenler bölümü bize iki fırsat sunarlar.

- Değişkenlere hatırlayacağınız isimler vererek onları daha sonra kolaylıkla hatırlayabilirsiniz
- Farklı bölümler için Visual Prolog dahilinde olmayan özel alan tanımlamaları yapıp bunları kullanabilirsiniz.

Değişkenler bölümünde vereceğimiz isimler önemlidir. Nedeni ise; bu değişkenlere vereceğimiz isimlere göre program kodları içinde rahatlıkla kaybolmadan dolaşabiliriz. Vereceğimiz değişken isimleri aynı zamanda programı açıklayıcı bir özeliğe de sahip olacaklardır. Diğer programa dillerinin aksine bu isimler Prolog için çok önemlidirler. Kullanıcın soracağı sorular da bu değişkenlere göre olacaklardır. Verilen cevaplara benzemelidir değişken isimleri de.

Ahmet 30 yaşında bir memurdur.

Yukarıdaki bilgiyi Visual Prolog'un anlayacağı dile çevirecek olursak aşağıdaki gibi bir satırla karşılaşırız:

person(symbol, symbol, integer)

Yazdığımız kodlar birçok durumda iyi çalışacaktır. Makinenin kolaylıkla cevap vereceği bir koddur. Ama çok uzun bir program yazdığımızı düşünelim. Ve bu satırı yazdıktan sonra 2 ay daha devam ettiğimizi programı yazmaya. Kodlar içinde, uygun, anlaşılır ifadeler kullanılmadığı taktirde kaybolmamız kaçınılmaz gibi görünüyor.

6.5.Goals(Hedefler) Bölümü

Visual Prolog programında hedefler ve kuralların yapıları birbirine benzer. Bir hedef ile kural arasında 2 fark vardır. Bunlar;

- Hedef yazılırken :- işareti kullanılmaz.
- Visual Prolog program derlendiğinde otomatikman hedefleri çalıştırır.

Visual Prolog'ta bir program çalışırken bir hedef hata verirse diğer programlama dilleri gibi program hata verir. Burada işleyişte bir fark yoktur.

7. EŞLEŞTİRME(UNIFICATION) VE GERİYE İZ SÜRME(BACKTRACKING)

Visual Prolog hedefleri gerçekleştirirken tek tek top to down adını verdiğimiz en üstten en alta kadar tarama yaparak gerçekleştirir. Sorgulama sırasında verilen hedef karşılaştırma yöntemiyle satır satır yapılmaktadır. Visual Prolog'taki terimlerin arama yöntemiyle bulunarak verilen tek hale getirilmesine eşleştirme(unification) denir. Eşleştirme yapılırken Visual Prolog otomatik olarak aşağıdaki işlemleri gerçekleştirir.

- İki atom eğer aynılarsa birleştirilir.
- İki rakam eğer aynı değere sahiplerse birleştirilir.
- İki yapı eğer aynı ada ve özelliğe sahiplerse birleştirilir.
- İki listesi eğer verileri aynıysa birleşir
- Bir değişken diğer bir değişkenin yerini alabilir. Bu işleme binding(bağlayıcı) denir.
- İki değişken bağlayıcı olarak birleşirler.
- İki string eğer aynı karakterleri içeriyorlarsa birleşirler.
- Bir string ve bir atom eğer aynı karakterlere sahiplerse birleşirler.

Bir Prolog programı derlendiğinde, aranan ifadeyi program içinde eşleştirme yaparak bulmaya çalışır. Program yazılan ifadeyi baştan sona kadar aramaya başlar. Aradığı hedefle eşleşen bir veri bulduğunda, bu veriyi serbest değişkenlere yükler ve hedefle aranan bilgi ayn hale gelir.

```
DOMAINS
title,author = symbol
pages = unsigned
PREDICATES
book(title, pages)
written_by(author, title)
long_novel(title)
CLAUSES
written_by(fleming, "DR NO").
written_by(melville, "MOBY DICK").
book("MOBY DICK", 250).
```

```
book("DR NO", 310).
long_novel(Title):-
written_by(_, Title),
book(Title, Length),
Length > 300.
```

Örneğimizde X ve Y serbest değişkenlerdir. Bulunana veriler serbst değişkenlere yüklenerek kullanıcıya aktarılacaktır. Prolog program derlendikten sonra hemen verileri taramaya başlar. Burada eşleştirme(unification) yöntemini kullanır ve kullanıcının istediği verileri bulmaya başlar.

Visual Prolog burada bir eşleştirme işlemi yapar ve X, Y değişkenlerine sırasıyla "fleming" ve "DR NO" değerleri atanır.

7.1.GERİYE İZ SÜRME(BACKTRACKING)

Prologta program yazımı esnasında sık sık belirli bir yolu izlemeniz gerekebilir. Eğer bu yol size geçerli bir sonuç vermiyorsa, alternatif yollar bulmak ve onları izlemek zorundasınız. Örneğin çocukken oynadığımız oyunlardan birini oynuyoruz ve bir yola saptık. Yolun sonunda aradığımız şeyin olacağını varsayarak yürüyoruz. Ama birden yolun çıkmaz bir sokak olduğunu gördük. Bu anda hemen geriye dönüp diğer yolları denememiz gerekiyor. Ve biz de geri dönerek diğer yollara bakıyoruz amacımıza ulaşabilmek için.

Bu şekilde tüm alternatif yolları deneyerek, oyuna devam ediyoruz ve uygun yolu bulduğumuzda amacımıza ulaşıyoruz ve programımız sonlanıyor. Aslında bu bir labirente benziyor. Labirentte kaybolduğumuz anda hemen denemeye başlarız yolları. Her biri yanlış çıkabilir. Burada önemli olan geçtiğimiz yoldan tekrar geçmemektir. Geçtiğimiz tüm yolları işaretleyip tekrar orda zaman kaybetmezsek amacımıza daha kolay ulaşabiliriz.

Visual Prolog oynadığımız bu oyunla aynı mantıkta çalışır. Geriye döner ve tekrar dener. Ama işaretleyerek, aynı yoldan tekrar tekrar geçmeden. Bu işleme **geriye iz sürme** veya **backtracking** diyoruz. Visual Prolog bir amaca yönelik sorgulama yaparken iki seçenekle karşılaşabilir. Bu iki seçenek arasında hemen bir tercih yapıp birine sapar veya dallanır. Ve buraya **backtracking point** adını verdiğimiz bir

belirteç koyarak o yerin hafızada kalmasını sağlar. Eğer Visual Prolog yanlış bir yola sapmışsa tekrar geriye dönerek backtracking point'e varır ve kaldığı yerden taramaya devam eder.

```
PREDICATES
likes(symbol,symbol)
tastes(symbol,symbol)
food(symbol)
CLAUSES
likes(bill,X):-
food(X),
tastes(X,good).
tastes(pizza,good).
tastes(brussels_sprouts,bad).
food(brussels_sprouts).
food(pizza).
```

Yukarıdaki program Bill'in ne sevdiğiyle ilgili yazılmış küçük bir programdır. Geriye iz sürmenin nasıl gerçekleştiğini görmek için bu amaca bakmalıyız.

```
likes(bill, What).
```

Burada unutmamamız gereken konu Visual Prolog'un bir amacı gerçekleştirmeye çalışırken, programın en başından başlar eşleştirmeye. Daha sonra alt satırlara doğru devam eder. Aramanın yapıldığı sırada Prolog her satıra, amaca, yorumlara ve kurallara bakarak eşleştirmelere bakar. Kendisine verilen sorgu, örneğin burada Bill'in neyi sevdiği soruluyor, gerçekleştiğinde sorgulamayı bırakır. Burada ilk bulunan **food** ilişkisidir. Burada food(x) sorgusuna verilecek cevap sayısı birden fazladır. Dolayısıyla Visual Prolog bu noktaya bir backtracking point koyarak programam devam eder. Burada konan backtracking point Visual Prolog programının yeni sorguya hangi noktadan başlayacağını belirtiyor.

Visual Prolog bir backtracking point'e geri döndüğünde bu noktadan sonra tüm değişkenleri serbest bırakır ve diğer sorgulama için eşleştirmeye devam eder. food(x)

çağrısı *brussels_sprouts* verisiyle eşleşecektir. Prolog bu sorguyu çözmeye çalışır kaldığı yerden devam ederek. Bu noktada food(pizza) verisini eşleştirmeye başlar veritabanındaki kayıtlarla. İstenilen sonuca ulaşıldığında Prolog aşağıdaki bilgiyi kullanıcıya gönderecektir.

What=pizza
1 Solution

7.2.Visual Prolog'un Sonuçlara Ulaşmak İçin Tarama Yapması

Visual Prolog bir probleme cevap ararken bulduğu ilk sonuca döndürmez kullanıcıya. Hemen hemen bütün sonuçları bulabilme yeteneğine sahiptir neredeyse. Şimdi de bir tenis kulübündeki yaşları ve isimleri bilinen çocukların katılacakları mini bir turnuva düzenleyelim.

```
DOMAINS
child = symbol
age = integer
PREDICATES
player(child, age)
CLAUSES
player(peter,9).
player(paul,10).
player(chris,9).
player(susan,9).
```

Öğrenciler arasında düzenlenecek olan turnuvayı Prolog Programında yazarak eşleştirmeleri verilen şartlara göre gerçekleştireceğiz. Burada her bir öğrenci çifti için iki oyun olacak. Burada amacımız dokuz yaşındaki öğrencilerin oluşturabilecekleri tüm muhtemel eşleştirmeleri bularak kullanıcıya sunmaktır.

```
goal
player(Person1, 9),
player(Person2, 9),
Person1 <> Person2.
```

Doğal programlama dillerinde yazılırken, *person1(age 9)*, *person2(age 9)* komutlarıyla kişiler bulunup buna göre eşleştirme yapılacaktı. Böylece birinci kişinin ikinciden farklılığı da gösterilecekti.

i. Prolog öncelikle *player(Person1,9)* komutuna bir cevap bulmaya çalışacaktır. Birinci amacına ulaştıktan sonra diğer açmalarını gerçekleştirmek için sıradakine geçecektir.

   player(Person2, 9)

   Ve son olarak üçüncü kişiye ait bilgileri sorgulamak için aşağıdaki sorgu çalıştıralacaktır.

   Person1 <> Person2

ii. Person1 ve Person2 Peter ismine çıktığı için Prolog hata verecektir. Dolayısıyla Prolog'un bu noktaya backtracking point koyması gerekir. Prolog bu yüzden işaretçiyi koyduktan sonra diğer amaçları gerçekleştirmeye devam eder.

iii. Kişiler bulunduktan sonra Prolog bu kişileri eşleştirirken değişkenlerin farklı olmasını kontrol eder. Aynı iki kişinin eşleşmesini engeller.

   Person1=peter, Person2=chris

   Person1=peter, Person2=susan

   Person1=chris, Person2=peter

   Person1=chris, Person2=susan

   Person1=susan, Person2=peter

   Person1=susan, Person2=chris

   6 Solutions

Prolog son olarak kullanıcıya eşleştirmeleri gösterir. Burada dikkat edeceğimiz konu Prolog'un kişileri eşleştirirken aynı kişileri eşleştirmemesidir. Bunun kontrolünü de backtracking dediğimiz geriye iz sürmeyle yapmaktadır.

8. VISUAL PROLOG PROGRAMLARININ ÇALIŞMA PRENSİBİ

Daha önce de belirtimiz gibi Prolog bir deklaratif programlama dilidir. Burada yapılan nesnelerin özellikleri, aralarındaki ilişkiler, kurallar, yapılar belirlendikten sonra bu veriler arasında eşleştirme mantığına göre yapılan aramalardır. Prolog tüm işlemleri yaparken Yapay Zeka ve Uzman Sistemler Mantığıyla çalışır.

Prosedürel programlamada bilgisayara adım adım ne yapacağı söylenir. Tüm komutlar yazılır. Nerede ne yapacağı, bir sorunla karşılaşırsa ne gibi yolar izleyeceği belirtilir. Deklaratif programlamada ise tanımlanan problem değil durumdur. Sistem durumu baz alarak problem çözümüne gider.

Prolog da bir prosedürel, mantıksal programlama dilidir. Prolog'un yapabileceği işlemleri aşağıdaki gibi sıralayabiliriz:

- Bir durumu tanımlayabilme,
- Bir cümlenin(clause) doğru ya da yanlış olduğunu anlayabilme.
- Cümle içinde verilen değişkenleri değerlere atayabilme,
- Arama yaparken geriye iz sürme yöntemini kullanarak(Backtracking) alternatif yollara başvurabilme.
- Değişkenler arasında belirlenen ilişkiler arasında kara verebilme,
- Değişkenlerin değerleri hakkında karar verebilme.(Mukakeme)

Prolog'ta önemli olan doğal hayattaki cümleleri Prolog programına anlatmaktır. Bizim ağzımızdan çıkan kelimeleri, cümleleri Prolog direkt olarak anlayamaz. Bu cümleleri bizim Prolog'un anlayacağı dile çevirmemiz gerekmektedir. Aşağıdaki örneklere bakacak olursak bunu görebiliriz:

- *Bütün çocuklar kısadır.*
  kisa(X):-cocuk(X).
- *Bütün erkek çocuklar arabaları sever.*
  sever(X,araba):-erkek(X),cocuk(X).
- *Bütün çocukların annesi vardır.*
  var(X,anne):-cocuk(X).
- *Sebzeyi hiçbir çocuk sevmez.*
  sevmez(X,Y):-sebze(Y), cocuk(X).
- *Çocuğunu döven öğretmeni hiçbir anne sevmez.*

sevmez(X,Y):-anne(X,Z),ogretmen(Y,Z),dover(Y,Z).

8.1. Visual Prolog'ta Cevap Nasıl Bulunur

Prolog hedefe ulaşmak için sisteme girilen durumları, değişkenleri, değişkenler arasındaki ilişkileri, kuralları bir bütün içinde kullanarak bir yargıya varır. Daha önce de belirttiğimiz gibi prolog bir problemin çözümünü değil içeriğini barındırır. Kullanıcı girilen verileri ve durumları göz önünde bulundurarak ilgili veriyi süzmektir.

Aşağıdaki örneğimizde cevabın nasıl bulunduğuna ilişkin önce kişiler arasındaki ilişkiler verilmiş, daha sonra bu ilişkilere ait kurallar verilmiştir. Ve Goal bölümünde tom ve pat arasındaki ata ilişkisi sorgulanmıştır. Burada sorgulama yapılırken satırlar tek tek yukarıdan aşağıya olmak üzere taranmaktadır. Satırlar taranırken Prolog aynı zamanda eşleştirme(matching) işlemini gerçekleştirir ve verilen kuralları sağlayan sonuçlarla işleme devam eder. dikkat edilecek olursa burada hemen evet veya hayır gibi bir cevap vermez prolog. Kurallar ve durumlar arasındaki ilişkileri tek tek denedikten sonra cevap verir.

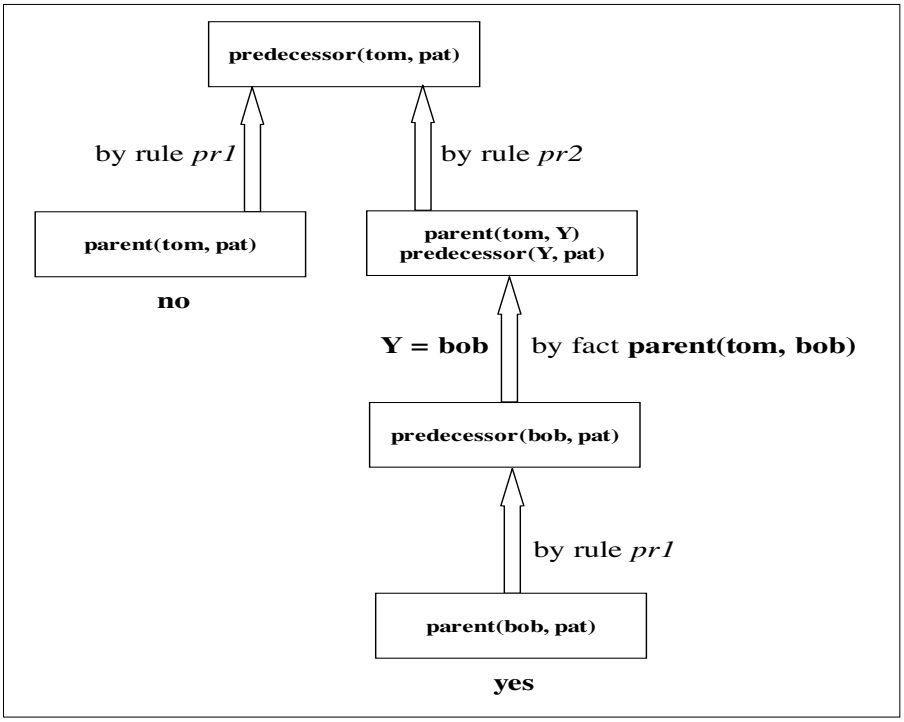
Programımız çalıştırıldığında ilk önce goal bölümündeki isteğe cevap verilmeye çalışır. Böyle bir veriye direk ulaşılamadığı için program diğer bölümlere dallanır hemen. Clauses bölümünde üç değişken bulunmaktadır. Serbest değişken adını verdiğimiz bu değişkenlere farklı değerler atanabilir. Buradaki seçim programcının kendisine kalmıştır. Bireyler arasındaki ebeveyn-çocuk ilişkisi belirlendikten sonra programcı tarafından kurallar belirlenmiştir.

Clauses

parent(pam, bob).
parent(tom, bob).
parent(tom, liz).
parent(bob, ann).
parent(bob, pat).
parent(pat,jim).
predecessor(X, Z) :- parent(X, Z).
predecessor(X, Z) :- parent(X, Y),predecessor(Y, Z).

Goal

predecessor(tom, pat).



Şekil 8.1. Prolog Çalışma Prensibi

9. VISUAL PROLOG'TA HAZIRLANMIŞ ÖRNEK PROGRAMLAR

Visual prolog birçoğumuzun saatler harcadığı zeka sorularını kısa bir zamanda çözebilir. Tabii burada önemli olan programı yazmaktır. Programcı programı yazarken durumları belirtecek, amaçları söyleyecek, ilişkileri oluşturacak yani mantıksal bölümü yine kendisi yapacaktır. Programa düşen ise kodları derlemek, ilişkileri sonuçlandırmak, programcının istediğin sonuçları döndürmektir.

9.1. Keçi, Lahana, Kurt ve Çiftçi

Programımız bir nehir kıyısında bulunana çiftçinin yanında bulunan keçi kurt ve lahanayı karşıya nasıl geçireceğiyle ilgilidir. Çiftçi karşıya yalnızca yanına keçi, kurt ve lahanadan birini alarak geçebilir. Sahip olduğu kayık büyük değildir.

Burada önemli olan hepsini karşıya sağ salim geçirebilmektir. Zira, çiftçi, lahanayı alıp karşıya geçmeye kalksa; kurt keçiyi yiyebilir, kurdu alıp karşıya geçse, keçi lahanayı yiyebilir. Bu yüzden dikkatli düşünmek zorundadır. Bizler de kodları yazmadan önce programın mantıksal bölümünü düşünüyoruz önce. Her birinin başlangıç ve hedef noktalarını tespit ediyoruz. Bir de unutmamamız gereken bir nokta, çiftçi her defasında kayığa binmektedir.

```
DOMAINS
  LOC   = dogu ; bati
  durum = durum(LOC çiftçi,LOC kurt,LOC keci,LOC lahana)
  PATH  = durum*

PREDICATES
  go(durum,durum)
  path(durum,durum,PATH,PATH)
  nondeterm move(durum,durum)
  opposite(LOC,LOC)
  nondeterm unsafe(durum)
```

```
 nondeterm member(durum,PATH)
 write_path(PATH)
 write_move(durum,durum)

GOAL
        go(durum(dogu,dogu,dogu,dogu),durum(bati,bati,bati,bati)),
        write("solved").

CLAUSES
 go(Startdurum,Goaldurum):-
        path(Startdurum,Goaldurum,[Startdurum],Path),
        write("A solution is:\n"),
        write_path(Path).

 path(Startdurum,Goaldurum,VisitedPath,Path):-
        move(Startdurum,Nextdurum),
        not( unsafe(Nextdurum) ),
        not( member(Nextdurum,VisitedPath) ),
        path( Nextdurum,Goaldurum,[Nextdurum|VisitedPath],Path),!.
 path(Goaldurum,Goaldurum,Path,Path).

 move(durum(X,X,G,C),durum(Y,Y,G,C)):-opposite(X,Y).
 move(durum(X,W,X,C),durum(Y,W,Y,C)):-opposite(X,Y).
 move(durum(X,W,G,X),durum(Y,W,G,Y)):-opposite(X,Y).
 move(durum(X,W,G,C),durum(Y,W,G,C)):-opposite(X,Y).

 opposite(dogu,bati).
 opposite(bati,dogu).

 unsafe( durum(F,X,X,_) ):- opposite(F,X),!.
 unsafe( durum(F,_,X,X) ):- opposite(F,X),!.

 member(X,[X|_]):-!.
 member(X,[_|L]):-member(X,L).
```

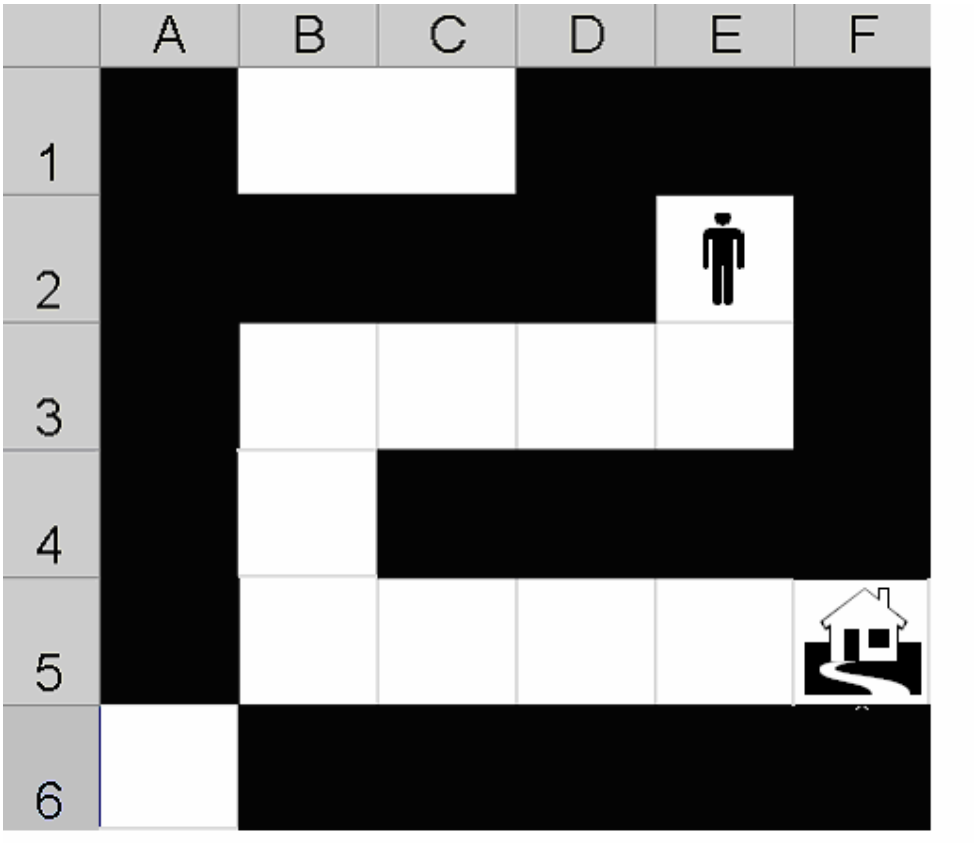
```
write_path( [H1,H2|T] ) :-
        write_move(H1,H2),
        write_path([H2|T]).
write_path( [] ).

write_move( durum(X,W,G,C), durum(Y,W,G,C) ) :-!,
  write("Çiftçi ",X,"dan ",Y,"ya dönüyor"),nl.
write_move( durum(X,X,G,C), durum(Y,Y,G,C) ) :-!,
  write("Çiftçi  kurdu ",X,"dan ",Y,"ya götürüyor"),nl.
write_move( durum(X,W,X,C), durum(Y,W,Y,C) ) :-!,
  write("Çiftçi  keçiyi ",X,"dan ",Y,"ya götürüyor"),nl.
write_move( durum(X,W,G,X), durum(Y,W,G,Y) ) :-!,
 write("Çiftçi  lahanayi ",X,"dan ",Y,"ya götürüyor"),nl.
```

Kodlara baktığımızda durumların tek tek tanımlandığını görmekteyiz. DOMAINS bölümünde kimin nerede duracağını kontrol edebilmek amacıyla *durum* kontrol birimi oluşturulmaktadır. Burada amaç, çiftçi hayvanlardan birini veya lahanayı yanına alıp karşıya geçtiğinde *doğu* veya *batı* diye nerede olduğunu bir değişken aracılığıyla tutmamızdır.

9.2. Labirent ve Yön Bulma Problemi

Yön bulma ve labirent problemlerinde, yolların birbirleriyle ilişkilerini sağlamak zorundayız. Sistem hangi yolların birbiriyle bağlantılı olduğunu bilmeli ve bu doğrultuda hareket etmelidir. Aynı satranç tahtasındaki karelerin birbirleriyle olan ilişkilerini belirlediğimiz gibi, labirentte yön bulurken de aynı işlem gerçekleştirilir. Labirentte esas olan iki hücredir. Bize düşen görev bu iki hücreyi buluşturmaktır. Tabii burada hücrelerden biri yani çıkış hücresi sabit durmaktadır. Bu hedef hücresidir. Hedef hücresi bizlerin varmak istediği noktadır. Bu noktaya varana kadar diğer hücremizi hareket ettiririz.

Burada yapmamız gerek diğer şey hücreler arası bağlantıdır. Örneğin labirentte çıkmaz yollara girebilme ihtimalimiz de var. Bu durumu kontrol etmek amacıyla, hücreler arası bağlantıyı kontrol ederiz. Hücreler arasında bağlantı varsa, zaten geçiş de vardır ve bir hücreden diğer hücreye geçebiliriz.



Şekil 9.1. Labirent Örneği

Örneğimizde 6*6 bir kare bulunmaktadır. Bizim amacımız E2 karesinde bulunan öğrencinin F5 karesinde bulunan evine gitmesini sağlamaktır. Dediğimiz gibi bunu sağlarken önce karelerin ilişkilerini belirlemeliyiz. Bunu belirlerken yapacağımız şey;

bagli(e2, e3).

bagli(e3, d3).

bagli(d3, c3).

bagli(c3, b3).

bagli(b3, b4).

bagli(b4, b5).

bagli(b5, c5).

bagli(c5, d5).

bagli(d5, d5).

kodlarını yazarak hangi hücrelerin bağlı olduğunu yani komşu olduğunu belirlemektir. Bundan sonra, aralarındaki kuralları belirlememiz gerekiyor.

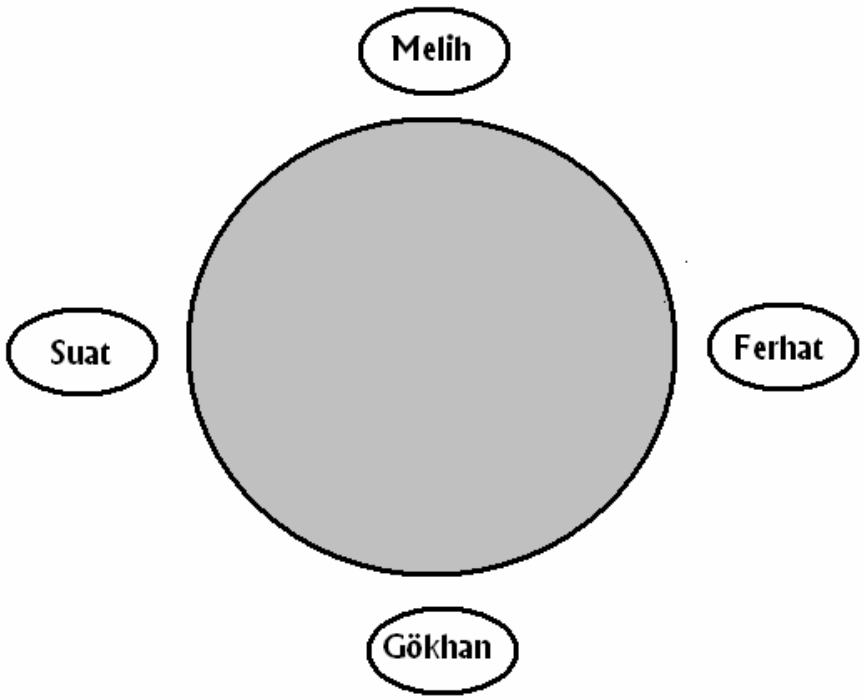
- **İki hücre (X ve Y) bağlıysa, geçiş vardır.**

  yolcikar(X, Y) :- bagli(X, Y) ; bagli(Y, X).

- **X, Y'ye geçiş olan herhangi bir Z noktasına bağlıysa, X'ten Y'ye geçiş vardır.**

  yolcikar(X, Y) :- bagli(X, Z) , yolcikar(Z, Y).

9.3. Yuvarlak Masa Problemi



9.2. Yuvarlak Masa Problemi

Yuvarlak masa problemi, masaya oturmuş kişilerin oturduğu düzenle alakalıdır. Kimin yanında kimin oturduğunu bilen sistem sorulan sorulara hemen cevap verebilir. Ve bu doğrultu masada değişiklikler yapabilir. Bizim masamızda dört kişi oturmaktadır. Bu dört kişi masaya otururken hep birilerinin yanına oturmaktadır. Örneğin Gökhan, Suat'ın sağında otururken Ferhat'ın solunda oturmaktadır. Suat ise, Gökhan'ın solunda otururken, Melih'in sağında oturmaktadır. Sistem bu bilgileri bilmek zorundadır. Sistem cevaplayacağı soruları bu bilgilere göre cevaplayacaktır. Öğrencileri masaya yerleştirirken bir kez yerleştirme yapılmalıdır. Ya da değişiklik yapıldıysa bundan sistemin de haberi olmalıdır. Şimdi bu bilgileri sisteme verelim:

saginda(gökhan, suat).

saginda(ferhat, gökhan).

saginda(melih, ferhat).

saginda(suat, melih).

solunda(suat, gökhan).

solunda(gökhan, ferhat).

solunda(ferhat, melih).

solunda(melih, suat).

Buradan belirli çıkarsamalara yani kurallara varılabilir. Örneğin; Gökhan, Suat'ın sağındaysa; Suat da Gökhan'ın aynı zamanda solundadır. Bunun yanında kişileri bu komşuluk ilişkilerine göre sıralayabiliriz. Örneğin; Gökhan Suat'ın sağındaysa, ve Ferhat Gökhan'ın sağındaysa, buradan çıkarılabilecek sonuç, Suat, Gökhan ve Ferhat'ın sırayla oturduklarıdır. Öğrenci sayısı arttıkça, bu sıralamalar da artacaktır doğal olarak.

10. SONUÇ VE ÖNERİLER

Zeki öğretim sistemleri, yapay zeka, uzman sistemler ve pedagojik formasyonun bir bütünüdür. Birinin eksikliği sistemin tam olarak çalışmamasına sebebiyet vereceğinden her bir bölüm çok önemlidir. Zeki öğretim sisteminin içeriği modüllerden oluşmaktadır. Her bir modülün işlevi tek tek oluşturulmakta ve aralarındaki ilişkiler sağlanmaktadır.

Zeki öğretim sistemlerinin ana teması öğrencidir. Sistem mükemmeliyeti öğrenci için tasarlanmasından geçmektedir biraz da. Öğrencinin yeterliliği, hazır bulunuşluk seviyesi, öğrenme stratejileri hep birlikte zeki öğretim sisteminin yapısını oluşturmaktadır. Sistemin ilk yaptığı iş bu değişkenleri belirleyip ona göre bir yol izlemektir. Burada yapay zeka devreye girmektedir. Kullanılan yazılımın denetlemesi doğrultusunda belirlenen öğrenci özellikleri programın akış diyagramını da belirleyecektir bir ölçüde. Programın akışı; öğrencinin özellikleri doğrultusunda olacaktır.

Zeki öğretim sisteminin sınırlılıkları da vardır. Sistem her ne kadar eksiksiz, öğrenci yapısına göre gözükse de tasarlanması bir hayli zaman alacak bir sistem olabilir. Tabii ki bu ihtiyaçların belirlenmesi doğrultusunda olacak bir planlamadır. Zeki Öğretim Sistemi, öğretim için sunulmuş iyi bir çözüm olmakla gerçeklenmesi zaman alan, karmaşık bir sistemdir. Planlama ve gerçeklenme sürecinde yapay zeka tekniklerinin, bilgisayar teknolojilerinin ve öğretim teknolojilerinin bir arada kullanılması gereklidir. (Funda Dağ, Kadir Erkan, Zeki Öğretim Sistemleri, ZÖS)

Eğitim-öğretim her ülkenin büyük kaynaklar ayırdığı, gerek personel gerek fiziki şartlar olarak her yıl yenilediği, yazılım, program gibi materyallerle geliştirildiği bir alandır dünyanın her ülkesinde. Eğitim-öğretimde gittikçe artan teknoloji kullanımı, bilinçli, öğrencinin öğrenmesini hızlandıracak, kolaylaştıracak, ona yardımcı olacak mikroeğitim adını da verdiğimiz sistemlerin gerekliliğini zorunlu kılmaktadır.

Artık öğrencinin izlediği değil bizzat içinde bulunduğu, yönettiğin, bir şeyler öğrendiği, müdahale ettiği yazılımlar daha ön plana çıkmıştır. Ülkemizde de bu tür yazılımların eksikliği duyulmaktadır. Özellikle fen ve matematik derslerinde

öğrencilerin yaşamış oldukları başarısızlıklar Zeki Öğretim Sistemleri kullanılarak bir ölçüde de olsa başarıya dönüştürülebilir. Bu sistemlerin bir diğer özelliği de sıkıcı gibi görünün bu derslerin öğrenciye sevdirilerek başarının arttırılmasını sağlamaktır. Sağlanan başarı, sistemin başarısı olduğu kadar aynı zamanda öğrencinin de başarısı olacaktır. Burada öğretmenin görevi de çok önemlidir.

Öğretim her ülkenin, her toplumun gelişmişliğinin göstergesi olan bir belirteçtir. Bu amaç doğrultusunda bir çok ülkede, gelişmiş öğretim sistemleri alanında, özellikle de Zeki Öğretim Sistemleri alanında çalışmalar yapılmaktadır. Ülkemizde de, diğer ülkelerde olduğu gibi, öğretim sistemleri alanındaki bu yönelimin fark edilmesi ve bu yönde etkin çalışmalar gerçekleştirilmesi gereklidir. (Funda Dağ, Kadir Erkan, Zeki Öğretim Sistemleri, ZÖS)

KAYNAKLAR

1. AKYÜZ, R. Ömür (1997) "Zihnin fiziği", Bilgisayar ve beyin, Nar yayınları
2. AMASYALI, M. Fatih(2006), PrologDers Notları
3. AYAR, Tacettin(2006), Prolog Ders Notları
4. AYDIN, Yavuz Selim(1998), Prolog Programlama Dili İle Makine Mühendisliği Alanında Uzman Sistemlerin Hazırlanması Teknikleri,Yüksek Lisans Tezi
5. BARR and E. Feigenbaum.(1982) The handbook of Artificial Intelligence
6. BİNGÖL, Canan A. (1997) "Öğrenme ve bellek", Bilgisayar ve beyin, Nar yayınları
7. BRNA, Paul(2001), Prolog Programming
8. ÇAKIR, Mustafa (1997), Bilgisayar Destekli Sozdizimi Calışması ve Prolog
9. DAĞ, Funda, ERKAN, Kadir(2004), Zeki Öğretim Sistemleri, Prolog Tabanlı Zeki Öğretim Sistemi (Zös), II. Bilgi Teknolojileri Kongresi
10. DOĞAN, Semra(2004), Mantıksal Programlama
11. FREEDMAN, David H. (1993) "Yeni nesil robotlar", Bilim teknik dergisi
12. GILLMOR, Steve (1998) "İnsan sesini algılayan bir uygulama", BYTE bilgisayar dergisi
13. GÖZKAN, Bülent (1997) "Bilgi, bilinç ve yapay zeka", Bilgisayar ve beyin, Nar yayınları
14. İNÖNÜ, M.Nazlı(1997) "Yapay zeka felsefesi", Bilgisayar ve beyin, Nar yayınları
15. KARAÇAY, Timur, Akıllı Veri Tabanları Yaratmaya Doğru
16. KOÇ, Nazım, KOCABAŞ, Şakir(2005), Prolog'un paralel mantık programlamaya genişletilmesi
17. ÖNDER, Hasan H.(2003), Uzaktan Eğitimde Bilgisayar Kullanımı ve Uzman Sistemler,
18. ÖZKAN, Murat Tolga, GÜLESİN, Mahmut(1999), Uzman Sistem Yaklaşımı ile Cıvata ve Dişli Çark Seçimi
19. PAÇACI, Görkem(2006), Ders Notları
20. TURBAN, Efraim(1991) Fundamentals of Management Science
21. Uğur, Aybars, Yapay Zeka Dersi, Ders Notları
22. WESLEY, Addison(2001), PROLOG Programming for Artificial Intelligence

ÖZGEÇMİŞ

Gökhan KARAOSMANOĞLU

Haziran, 2007

KİMLİK BİLGİLERİ

Gökhan Karaosmanoğlu 1979 İzmir doğumludur. Bekardır

ADRES (İş)

Insirah Caddesi Manolya Sokak No:16 Bebek - Besiktas / İstanbul,Türkiye

İLETİŞİM

Tel : 90 (212) 263 61 47
Faks : 90 (212) 257 01 68
E-mail : gkaraosmanoglu@hotmail.com

EĞİTİM

1996 **İlköğretim**
Güzelcehisar İlköğretim Okulu, Beykoz-İstanbul,Türkiye

2000 **Lise**

Kağıthane Profilo Anadolu Teknik Lisesi, Bilgisayar Bölümü
Kağıthane/İstanbul,Türkiye

2004 **Lisans**
Çanakkale 18 Mart Üniversitesi, Eğitim Fakültesi, Bilgisayar ve Öğretim Teknolojileri Eğitimi Bölümü, Çanakkale, Türkiye

2007 **Yüksek Lisans**
Haliç Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Bölümü, İstanbul, Türkiye(Devam Etmektedir)

ÇALIŞMA HAYATI

2004-2006 Bilgisayar Öğretmeni
Cengizhan İlköğretim Okulu
Kağıthane, İstanbul

2004-2006 Bilgisayar Öğretmeni
Tevfik Fikret İlköğretim Okulu
Bebek-Beşiktaş, İstanbul

2004-2007 Bilgisayar Öğretmeni
Sarıyer Belediyesi Bilgisayar Kursları
Yetişkin Bilgisayar Eğitimi

YABANCI DİLLER

Ingilizce ve Fransızca biliyor. İlköğretim düzeyinde ingilizce dersleri veriyor

İLGİ ALANLARI

Satranç İlköğretim okullarında satranç eğitimi veriyor, lisanslı satranç hakemi ve eğitmeni.

BDÖ Bilgisayar eğitiminin ilköğretim okullarında iyileştirilmesi ve diğer derslerde kullanılmasıyla ilgili araştırmalar yapıyor.

Linguistic Dilbilimi ve dilin daha kolay nasıl öğretilebileceği konusunda araştırmalar yapıyor.

Müzik Amatör olarak gitar ve bağlama çalıyor.